LASER 128/128EX

PERSONAL COMPUTER

- USER'S GUIDE
- BASIC MANUAL

NON-DISCLOSURE AGREEMENT
AND
REGISTRATION FORM

The party below agrees that it is receiving a copy of Microsoft BASIC for use on a single computer only, as designated on this non-disclosure agreement. The party agrees that all copies will be strictly safeguarded against disclosure to or use by persons not authorized by Video Technology Computers, Ltd. to use Microsoft BASIC, and that the location of all copies will be reported to Video Technology Computers, Ltd at Video Technology Computers, Ltd's request. The party agrees that copying or unauthorized disclosure will cause great damage to Video Technology Computers, Ltd. and the damage is far greater than the value of the copies involved. The party agrees that this agreement shall inure to the benefit of any third party holding any right, title or interest in Microsoft BASIC or any software from which it was derived.

Purchased From:

Company

Address

Phone

Purchased By: (Distributor)

Name

Address

Phone

Date

For Use On:

Model

Serial #

Software Product

Purchased By: (Dealer)

Name

Company

Address

Phone

Date

Purchased By: (End-User)

Name

Company

Address

Phone

Date

NOTE: The Non-Disclosure Agreement MUST be signed by Party purchasing product directly from Video Technology Computers Ltd.. No product will be shipped without signed agreement. It is the responsibility of Distributor and/or Dealer to transfer ownership to appropriate party.

Completed form should be returned to the following address:

(For product sold in all countries except U.S.A.)
VIDEO TECHNOLOGY
COMPUTERS LTD.
23/F., Tai Ping Ind. Centre, Blk. 1,
57 Ting Kok Rd., Nam Hang, Tai Po,
N.T., Hong Kong.

(For U.S.A. only)
VIDEO TECHNOLOGY
COMPUTERS INC.
2633 Greenleaf,
Elk Grove Village,
IL 60007 U.S.A.

FIRST EDITION - 19877

The following is applicable to U.S.A. FCC class B version only

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, with the specifications to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:
Reorient the receiving antenna
Relocate the computer with respect to the receiver
Move the computer away from the receiver
Plug the computer into a different outlet so that computer and receiver are on different branch circuits.
If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:
"How to Identify and Resolve Radio-TV Interference Problems".
This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock No. 004-000-00345-4.

Use peripheral cables supplied by Video Technology. These cables have special shielding to prohibit RF interference.

PRECAUTIONS

1. **POWER UP PRECAUTION**
   The computer hardware will not lose memory immediately after the power is turned off. So the user should wait for a very short moment to turn on the computer after the unit has been powered down. Otherwise the computer will not cold-start properly.

2. **SETTING OF 40/80 COLUMN SWITCH**
   Some software needs this switch to be set to the 80 column position. Otherwise, the program may break out. Before running your software, please check that the switch is set to the required position.

3. **SETTING OF ALT/STD SWITCH**
   There is a keyboard layout switch on the back panel of the computer. Setting the switch to the STD position will give characters displayed as shown on the keytops. Setting the switch to the ALT position will give the DVORAK keyboard layout. E.g. typing the key with "R" on it will give a "P" character.

4. **COMPLIANCE WITH FCC REGULATIONS**
   The Expansion Connector on the left side of the computer is for connecting an optional FCC approved expansion box supplied by Video Technology Computers Ltd. Customers are advised to use the FCC approved peripherals and cables (e.g. external drive, joystick, serial printer cable, parallel printer cable, monitor cable, RGB cables etc.) supplied by Video Technology Computers Ltd. Failure to do so may violate the FCC regulations for a class B computing device.

(This page is intentionally left blank)

TABLE OF CONTENTS

PART II
BASIC MANUAL

APPENDIX

PART I

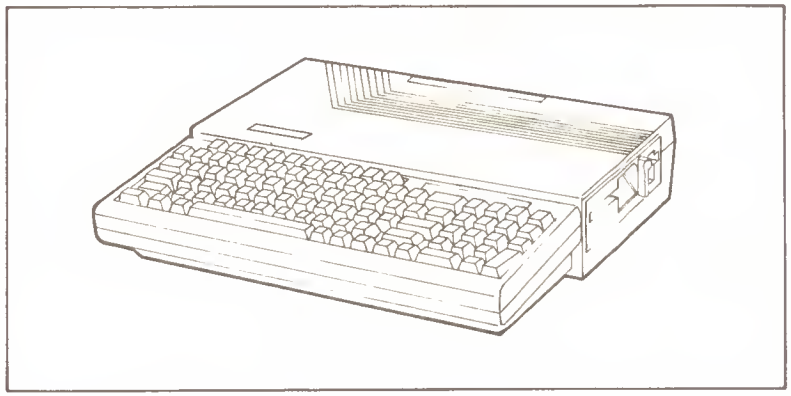
USER'S GUIDE

CHAPTER 1

A FIRST LOOK

1. A FIRST LOOK

Congratulations on your purchasc! You arc now thc proud owner of a powcrful and easy-to-usc computcr which is suitable for applications ranging from arcadc gamcs, pcrsonal computing and cducational aid up to scicntific and busincss data proccssing.



Figurc 1-1 Thc LASER 128/128EX

1.1 Features of the LASER 128/128EX

Your computcr is dcsigned for tomorrow as wcll as for today.

First of all, the computer has built-in interfaces for most of the common pcripherals so that you don't have to purchase expcnsive intcrface cards for deviccs such as disk drives or printcrs.

Secondly, it can easily be expandcd to accommodate more memory as well as Input/Output deviccs so that the computer systcm can be configurcd to suit all kinds of applications that you may cvcr think of.

Most important of all, the computcr can run almost all of the software writtcn for thc Apple® IIe computer so that as soon as you sct up the computer system, you have access to one of the world's largest softwarc library instantly!

The computer comes in two versions: The "LASER 128" & the "LASER 128EX". Thc LASER 128EX carries more fcatures ovcr the LASER 128 which make it evcn morc powerful. The following is a brief summary of some of the features of both versions of the computer:

- The 65C02 CPU (Central Proccssing Unit) used is an enhanced version of thc 6502 CPU having a larger instruction set and more addressing modes.

- In the LASER 128, the CPU runs at 1 MHz. In the LASER 128EX, thc CPU clock is software-sclectablc to be 1 MHz, 2.3 MHz or 3.6 MHz so that program execution can be spceded up at will while compatibility with the "standard" computer can still be maintaincd. (SEE APPENDIX A.)

- Built-in 32 K-byte ROM contains thc Microsoft® BASIC interpreter and software drivers for the various built-in Input/Output devices and interfaces.

- Built-in 128 K-byte system RAM. In the LASER 128EX, additional 64K-byte video RAM is built-in.

- Up to 1 M-byte expansion RAM can be added to the LASER 128 by plugging in an optional Memory Expansion Card. In the LASER 128EX, this card has already been included and all you need is to install supplementary RAM chips (see APPENDIX B.)

- The keyboard contains 90 step-sculptured keys including function keys, cursor-control keys, screen-editing keys and a separate numeric keypad.

- The keyboard layout is switch-selectable to be "QWERTY" or "DVORAK"*.

- Built-in speaker with volume control for sound generation.

- An earphone jack is provided for connecting an earphone or other audio output devices.

- Both 40-column and 80-column text displays are supported.

- Four graphics modes are available including:

  - Low-resolution graphics (40H x 48V, 16 colors)

  - Medium-resolution graphics (80H x 48V, 16 colors)

  - High-resolution graphics (280H x 192V, 6 colors)

  - Double-high-resolution graphics (560H x 192V, 16 colors)

* See p.I-26.

- Supports mixed graphics/text display for any of the four graphics modes.

- NTSC or PAL standard composite video output for color/monochrome video monitor or TV via a RF modulator.

- Supports versatile video display devices such as LCD panel and RGB monitor.

- Built-in 5.25" disk drive.

- Supports an external 5.25" or 3.5" disk drive.

- Supports Centronics-type parallel printers.

- Supports serial printers via built-in RS232C interface.

- Supports modems or other serial communication devices via built-in RS232C interface.

- Supports input devices such as joysticks, paddles and mouse.

- An expansion connector for plugging an optional FCC approved expansion box in which up to two peripheral interface cards can be installed.

1.2 The purpose of this manual

This manual is not intended to be a textbook of computer architecture or computer programming. There are already a lot of well-written books on those subjects. Instead, it is a user's guide geared to the first-time computer users. The materials which you will find in this manual include:

- Functional description of each part of the LASER 128/128Ex.
- Installation of the computer system
- Basic operations of the LASER 128/128EX.

As an ordinary user, you do not need to know anything about the internal workings of the computer (although that might be an advantage) before you can use it. Using the computer can be as simple as inserting an application program diskette into the built-in disk drive and turning it on. The rest of the work is done by the application program and is therefore transparent to the user.

However, if you are a programmer, you may also want to know the hardware and firmware details of the LASER 128/128EX in order to utilize its features efficiently. Those information can be found in the *Technical Reference Manual* available separately.

CHAPTER 2

GETTING FAMILIAR WITH THE LASER 128/128EX

2. GETTING FAMILIAR WITH THE LASER 128/128EX

Before you begin to work with your computer, you should learn something about the function of each of its components:

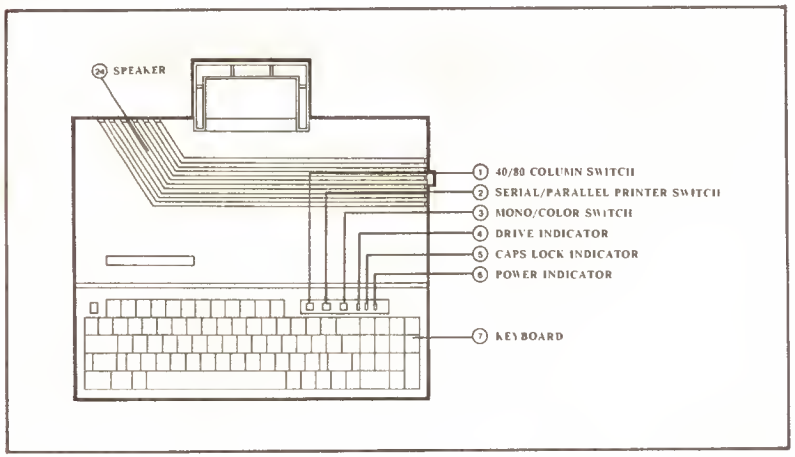


Figure 2-1 Top view of the LASER 128/128EX

(1) The 40/80 COLUMN SWITCH is for enabling/disabling 80-column text display. If you are using a TV or a composite color monitor, 80-column text may be difficult to be read. In this case, you can throw the switch to the "40-column" position so that text will only be displayed in 40-column format. If it is thrown to the "80-column" position, both 40-column and 80-column text can be displayed.

(2) The SERIAL/PARALLEL PRINTER SWITCH selects either the serial printer or the parallel printer connected to the computer as the printing device.

(3) The MONO/COLOR SWITCH is for enabling/disabling color video outputs. If you are using a color monitor or TV, throw the switch to the "COLOR" position in order to display graphics in color. If instead you are using a monochrome monitor or black-and-white TV set, we suggest you set the switch to the "MONO" position in order to have a clear graphics display.

(4) The DRIVE INDICATOR is lit up whenever the built-in disk drive is working. As a general rule, you should NOT insert or remove a diskette when a disk drive is turned on.

(5) The CAPS LOCK INDICATOR is lit up whenever the CAPS LOCK function of the keyboard is turned on. Pressing the CAPS LOCK key turns the CAPS LOCK function on and off alternately. When turned on, any letter typed on the keyboard will be displayed as a capital letter.

(6) The POWER INDICATOR is lit up whenever the computer is turned on. It provides an easy way to check whether the power supply is working properly or not.

(7) The KEYBOARD is, of course, for entering information into the computer. It contains 90 keys, including the numeric keypad on the right.

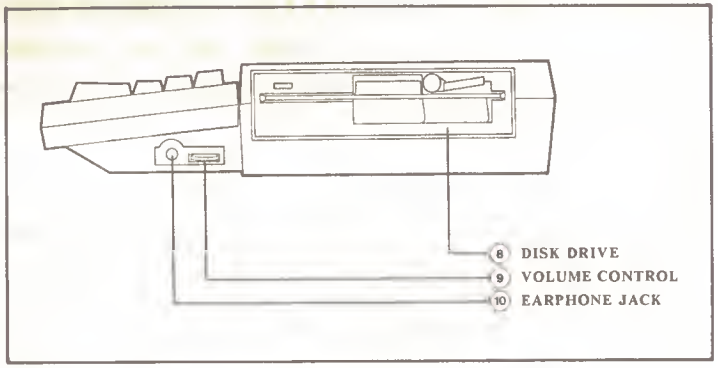


Figure 2-2 Side view (right) of the LASER 128/128EX

(8) The DISK DRIVE is the gadget that reads the programs or data recorded on a floppy disk into the computer. You may also store new information on a floppy disk using the disk drive. For example, if you are working with a word-processor, the text of the reports or letters you wrote can be stored permanently on a floppy disk. The built-in disk drive is a 5.25" drive.

(9) The VOLUME CONTROL located by the side of the earphone jack controls the volume of the sound that comes out of either the speaker or an earphone that is plugged into the earphone jack.

(10) The EARPHONE JACK located on the right side of the computer is for connecting an earphone. If an earphone is plugged into the jack, the audio outputs will be directed to it instead of the built-in speaker.

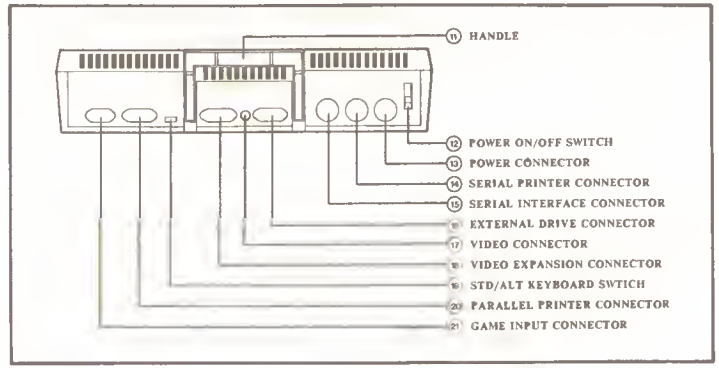


Figure 2-3 Back panel of the LASER 128/128EX

(11) The HANDLE provides a convenient way of carrying the LASER 128/128EX. In addition, it can also be used as a prop for the computer so as to give the keyboard a better typing angle and air ventilation.

(12) The POWER ON/OFF SWITCH disconnects the computer from the external power supply when it is thrown to the "OFF" position. Throwing it to the "ON" position, i.e. flipping it up, turns the computer on.

(13) The POWER CONNECTOR is where the external +17V DC power supply (usually the AC power adaptor) goes. Refer to Figure C-1 in APPENDIX C for pin assignment.

(14) The SERIAL PRINTER CONNECTOR is for connecting a serial printer with RS232C interface. The rate of data transfer is software-selectable from 110 to 19200 baud. The pin assignment is illustrated in Figure C-2.

(15) The SERIAL INTERFACE CONNECTOR is for connecting serial communication devices with RS232C interface (e.g. modem). The rate of data transfer is software-selectable from 110 to 19200 baud. The pin assignment is shown in Figure C-3.

(16) The EXTERNAL DRIVE CONNECTOR is for connecting an external disk drive. The drives may be LASER FD100c compatible 5.25" drives, LASER FD356 compatible or Unidisk™ compatible 3.5" drives. Many application programs can make use of the second disk drive to reduce disk swapping so that the program can run faster. The pin assignment is shown in Figure C-4.

(17) The VIDEO CONNECTOR is for connecting a NTSC/PAL standard monochrome/color composite video monitor. The pin assignment is shown in Figure C-5.

(18) The VIDEO EXPANSION CONNECTOR located by the side of the video connector is for connecting other sophisticated display devices such as a flat-panel LCD display, a TV interface or a RGB monitor. In PERITEL (non-U.S.A.) versions, it also provides the PERITEL video signals. The pin assignment is shown in Figure C-6.

(19) The STD/ALT KEYBOARD SWITCH allows you to select the STanDard "QWERTY" keyboard layout or the ALTernate "DVORAK" layout.

(20) The PARALLEL PRINTER CONNECTOR is for connecting a parallel printer with Centronics interface. The pin assignment is shown in Figure C-7.

(21) The GAME INPUT CONNECTOR is for connecting joysticks, paddles or a mouse which are used by some application programs as input devices instead of the keyboard. The pin assignment is shown in Figure C-8.

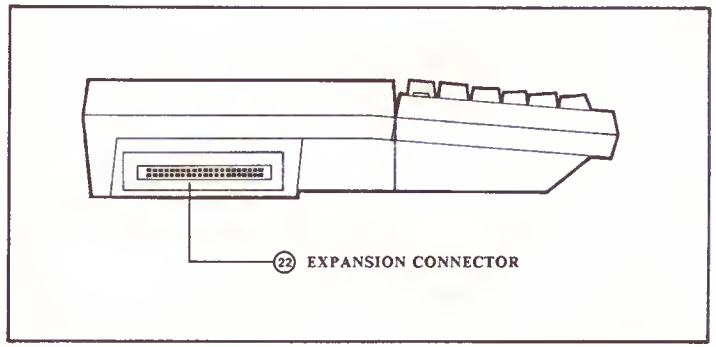


Figure 2-4 Side view (left) of the LASER 128/128EX

(22) The EXPANSION CONNECTOR on the left side of the computer is for connecting an optional FCC approved expansion box supplied by Video Technology Computers Ltd. Up to two peripheral cards can be installed in the expansion box. Besides providing mechanical protection to the interface cards, the expansion box, with its own power supply, also helps to ease the drain on the mother-board power supply by the peripheral cards. The pin assignment is shown in Figure C-9.

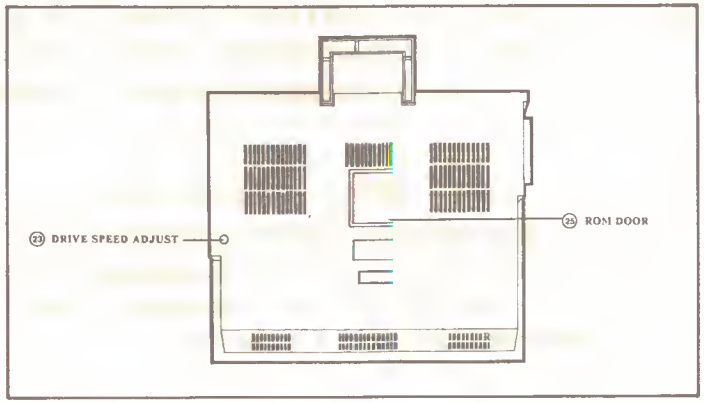


Figure 2-5 Bottom view of the LASER 128/128EX

(23) The DRIVE SPEED ADJUST located at the bottom of the computer allws you to adjust the rotational speed of the buil-in disk drive with a small screw-driver. Normaly, you should NOT adjust the drive speed. Hovever, for some drive speed critical application pograms, you may need to trim the drive speed slihtly in order to run them successfully.

(24) The SPEAKER inside the computer is used by some programs for geneating sound effects. Immediately after the comuter is turned on, the speaker will make a short "bep" sound to alert the user.

(25) The ROM DOOR allows the user to gain access to part of the printed circuit board where a ROM chip and three switches are located.

(26) If you open the ROM door by loosening the two screws which mount it on the bottom cabinet, you'll see something like this:

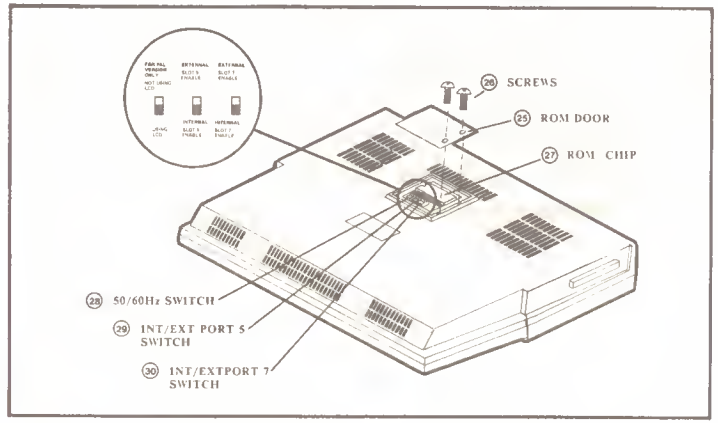


Figure 2-6 Underneath the ROM door

(27) The ROM chip contains the Microsoft® BASIC interpreter and I/O drivers. It is mounted on a 28-pin IC socket soldered onto the bottom of the PCB.

(28) The 50/60Hz SWITCH is for selecting the vertical field rate of the video display. For the NTSC version this switch is not necessary so it is not included. For the PAL or PERITEL versions, it should be set to the 50Hz position. However if a LCD panel from Apple® is used, this switch should be set to 60Hz for synchronising display.

(29) The INT/EXT PORT 5 SWITCH selects either the INTernal expansion memory interface or the EXTernal peripheral interface card installed in the optional expansion box as the I/O device at port 5 of the computer. An I/O (Input/Output) port is the channel through which a computer communicates with the real world. There are a total of seven I/O ports in the computer.

(30) The INT/EXT PORT 7 SWITCH selects either the INTernal 3.5" disk drive interface or the EXTernal peripheral interface card plugged in the optional expansion box as the I/O device at port 7 of the computer.

CHAPTER 3

GETTING STARTED

3. GETTING STARTED

At this point, you should have some basic ideas of the LASER 128/128EX and its capabilities so that you may start to work with it actually. To set up the computer system, you need the following items:

- The Computer main unit
- The AC power adaptor included
- The video cable included
- A video monitor, or a TV with a TV interface

3.1 Connecting the AC power adaptor

The computer main unit will operate with a power supply from +13V DC to +18.7V DC. For greater portability, the AC power adaptor is separated from the main unit and the power source can also be a battery pack.

To connect the power adaptor, locate the power ON/OFF switch and the power connector on the back panel of the main unit. Throw the switch to the "OFF" position. Plug the DC power plug of the AC power adaptor into the power connector of the computer. Finally, plug the AC power plug into the wall outlet. Figure 3-1 shows details of the procedures.

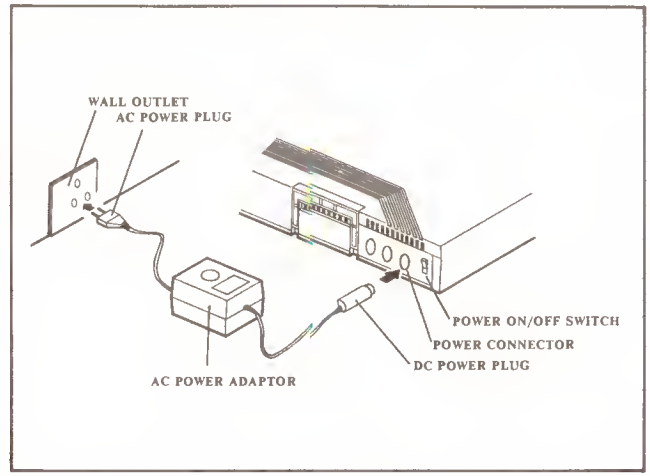


Figure 3-1 Connecting the AC power adaptor

3.2 Connecting the video monitor/TV

If you are using a monochrome/color composite video monitor, plug one end of the video cable into the video connector on the back panel of your computer and the other end into the connector of the monitor as shown in Figure 3-2.

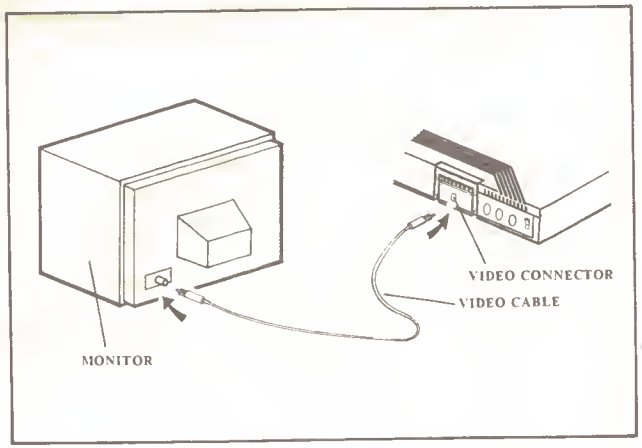


Figure 3-2 Connecting a video monitor

If instead you are using a TV with the optional TV interface, first connect the interface to your computer's video expansion connector as in Figure 3-3. Then connect the interface to your TV's antenna leads, following the instructions that come with your TV interface.

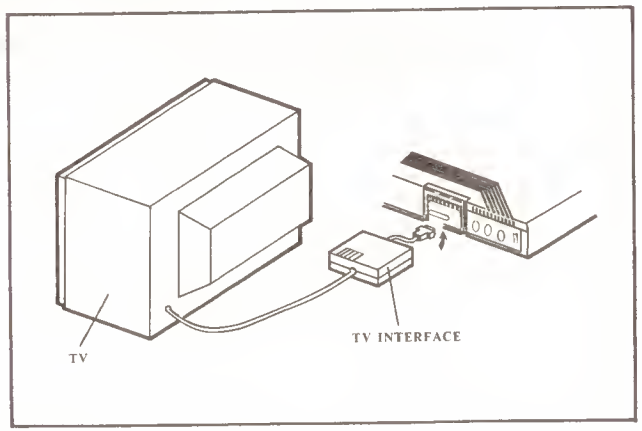


Figure 3-3 Connecting a TV

3.3 Starting up the computer system

Now you have set up the computer system in its simplest configuration. To start it up, here are the procedures to follow:

1) OPEN THE DRIVE DOOR by turning the lever in the anti-clockwise direction until it rests in the horizontal position as shown in Figure 3-4.

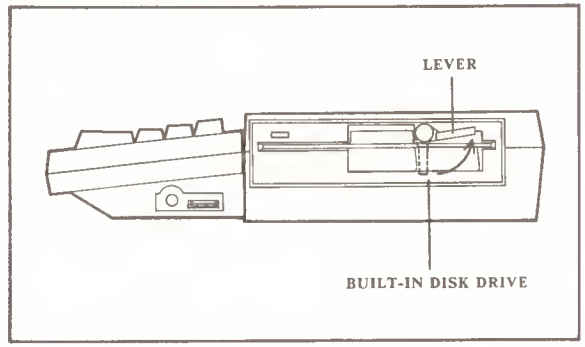


Figure 3-4 Opening the drive door

2) If you are not using an auto-startup program diskette, skip to step 4. Otherwise, INSERT THE DISKETTE into the computer's built-in disk drive gently, with the labelled side of the diskette facing upwards and the end with an oval-shape opening entering first. See Figure 3-5.

WARNING: Inscrting a diskette in the wrong orientation may damage the diskette and/or the disk drive.

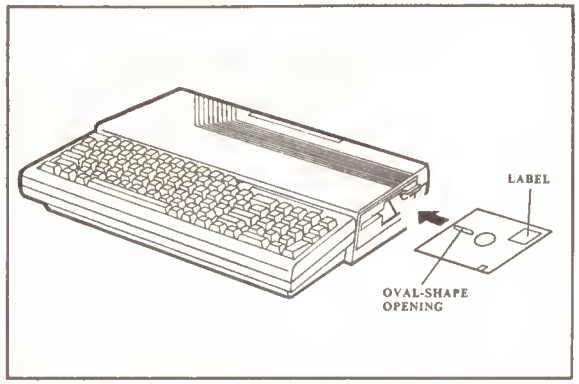


Figure 3-5 Inserting a diskette

3) CLOSE THE DRIVE DOOR by turning the lever in the clockwise direction until it rests in the vertical position as in Figure 3-6.

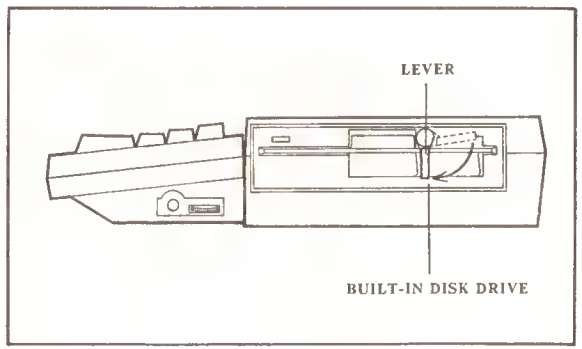


Figure 3-6 Closing the drive door

4) TURN ON THE VIDEO MONITOR/TV first since they take a few seconds to warm up.

5) TURN ON THE COMPUTER by throwing the power ON/OFF switch to the "ON" position. The following things will then happen:

- The power indicator is lit up. If it is not, check whether the power adaptor is properly connected to the computer and the wall outlet or not.
- The speaker makes a short "bcep" sound. If not, check whcther the volume control is properly adjusted or not.
- There are words displayed on the top of the screen. If not, check whether the monitor is properly connccted to the computer and whether the brightness control knob of the monitor is in an appropriate position.
- The drive indicator is lit up and the built-in disk drive makes a few "ratchety" noises for a momcnt.

If an auto-startup program disk is inserted in the internal drive, the program will be loadcd and run automatically. You should then refer to the manual of that application program for further instructions.

If instead you want to use the computer without a program disk, you should then hold down the "CTRL" key and press the "RESET" button once. The following things will happen:

- A "beep" sound is made by the speaker.
- The drive indicator goes off and the disk drive stops running.
- The prompt character "]" and a flashing "checker-board" cursor are displayed on the bottom of the screen to inform you that you are now in BASIC command level.

You may then type in any valid BASIC commands or statements. (However, you can't save Basic programs on disk if you have not started up the computer with a disk containing operating system.) Refer to PART II of this manual for a detailed description of BASIC commands.

CHAPTER 4

THE KEYBOARD

4. THE KEYBOARD

The keyboard is the most frequently used input device of the computer. Most of your instructions and data are entered into the computer through the keyboard. As a result, a well-designed and easy-to-use keyboard is essential in the standpoint of a user. The following is a brief summary of some of the features of the keyboard of LASER 128/128EX:

- The specially-shaped keytops and comfortable key touch make typing easy.
- Auto-repeat function reduces typing effort.
- The keys are arranged in such a way that the most commonly used keys, like "SHIFT" and "RETURN" , can be located easily.
- Switch-selectable "QWERTY" or "DVORAK"* keyboard layout.
- Separate numeric keypad facilitates numeric data entry.
- Special keys for cursor-control and screen-editing.
- 10 predefined function keys facilitates entry of commonly used control-characters.
- Two key rollover helps speed typing.
  i.e. A second key can be pressed without releasing the first key pressed.

* Applicable only in U.S.A., different keyboard layouts are used as substitutes in different countries.

4.1 Typewriter keys

These are the keys which you can find on the keyboard of a typewriter. They include the alphabetic and punctuational keys, the "SHIFT" keys, the "CAPS LOCK" key, the "TAB" key and the "DELETE" key. The arrangement of these keys resembles that of a typewriter so that your typing skills can be applied on the computer.

There are two possible layouts for the typewriter keys, selected by the STD/ALT keyboard switch located on the back panel of the computer. When the switch is thrown to the "STD" position, you get the standard "QWERTY" layout shown in Figure 4-1.

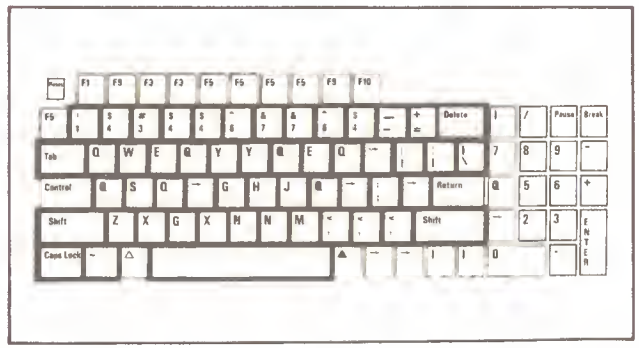


Figure 4-1 "QWERTY" keyboard layout.

When the switch is thrown to the "ALT" position, you get the alternate "DVORAK" layout or a layout of some other countries. Figure 4-2 shows the "DVORAK" layout.

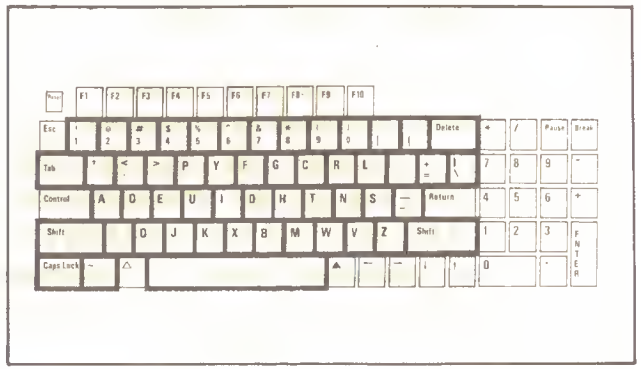


Figure 4-2 "DVORAK" keyboard layout

The "SHIFT" keys are functionally equivalent to those of a typewriter. If you hold down the "SHIFT" key while typing a letter key ("A" to "Z"), you will always get a capital letter.

If instead you are typing a key having two symbols on it, you will get the upper symbol shown on the keytop. For example, typing the "=" key alone gives you an equal sign while typing it with the "SHIFT" key held down will give you a plus sign.

The "CAPS LOCK" key is similar to the SHIFT LOCK key of a typewriter except that it will only affect the letter keys. Pressing the key turns the CAPS LOCK indicator on and off alternately.

When the CAPS LOCK indicator is on, typing any letter key will give you a capital letter, regardless of whether the "SHIFT" key is held down or not.

When the CAPS LOCK indicator is off, typing any letter key alone, i.e. without holding down the "SHIFT" key, will give you a lowercase letter.

The "TAB" key is used by many application programs (e.g. word-processors) for moving the typing position to the next "tab-stop" position on the screen.

The "DELETE" key is used by many application programs for correcting typing mistakes. Pressing the key causes the character at the current typing position on the screen (usually indicated by a flashing character called the cursor) to be erased.

REMARK: The "TAB" key and "DELETE" key work properly in some but not all programs. For instance, the built-in BASIC interpreter cannot recognize these keys.

4.2 Cursor-control keys

These include the left-arrow " ← " key, right-arrow " → " key, up-arrow " ↑ " key and down-arrow " ↓ " key. (See Figure 4-3) Typing one of these keys in the BASIC command level causes the current typing position to be moved in the direction pointed to by the arrow-head, with the exception of the up-arrow " ↑ " key which has to be pressed subsequent to the Escape "ESC" key. These keys provide a convenient way of editing the display. Different programs use the arrow keys in different ways. You should check the user's manual of each application program for details.

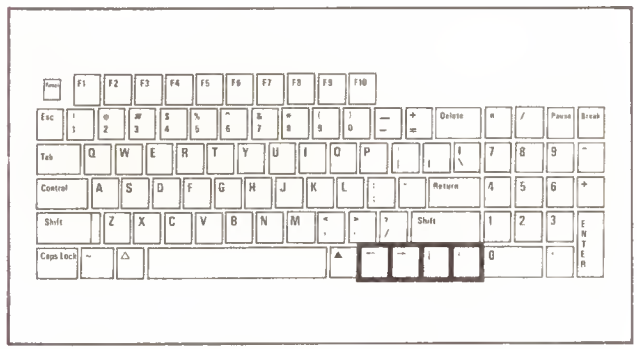


Figure 4-3 Cursor-control keys

4.3 Command keys

Figure 4-4 shows the command keys which are primarily used for telling the computer to perform some immediate actions.

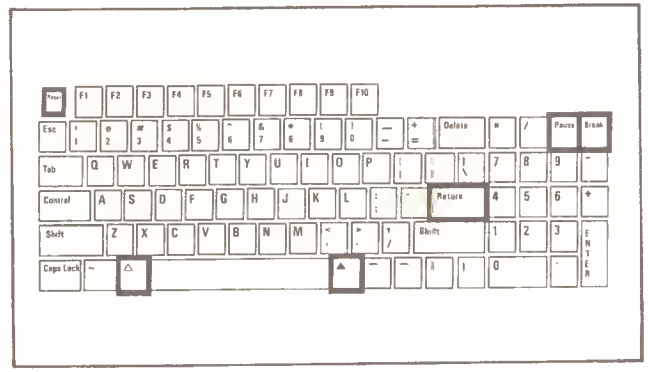


Figure 4-4 Command keys

The "RETURN" key serves two purposes. First of all, it acts as the carriage-return of a typewriter. Pressing it ends the current line you're typing and moves the cursor to the start of the next line. Secondly, it tells your computer that you have finished typing this line and it may now process it. The lines you type for a program will usually be of variable length. As a result, pressing "RETURN" is necessary when you have done typing a line.

The "PAUSE" key is for temporarily stopping the display. For instance, if a large piece of text is being displayed on the screen, the information may move off the screen faster than you can read.

In this case, you can press the "PAUSE" key. The display will stop as soon as it finishes printing the current line. To resume the display, press any key on the keyboard.

The "BREAK" key is for aborting the execution off a BASIC program. If you get lost in a BASIC program and do not know the way out, pressing the "BREAK" key will break you out and return you to the BASIC command level. This is a fairly drastic step and may cause some data to be lost.

The "ESC" key is often used by application programs as means to ESCape out of whatever you are doing, just like the function of the "BREAK" key in a BASIC program.

In some programs, it is also used as a "lead-in" character for special multi-character commands. For instance, pressing the "ESC" key followed by the "I" key (abbreviated as ESC I) in the BASIC command level causes the cursor to move up one line on the screen.

The "RESET" key represents a very drastic step. Pressing it while holding down the "CTRL" key (abbreviated as CTRL-RESET) causes the computer to abort whatever it is doing and restart all over again. This often causes the program and data in memory to be lost as well.

In some cases, the computer may get stuck and stop running.

In this case, pressing "CTRL-RESET" is the only way to bring you out of this situation. The two-key sequence prevents you from resetting the computer accidentally.

The hollow-triangle " △ " and solid-triangle " ▲ " keys are special purposed keys which correspond to the open-apple and closed-apple keys on Apple® IIe/IIc computers. The computer can tell whether they are pressed or not, independent of the status of the other keys on the keyboard. The use of these keys differs from program to program.

4.4 The "CTRL" key

The function of the "CTRL" key is similar to that of the "SHIFT" keys. If you hold down the "CTRL" key while pressing another key, a special control-character will be generated. These control-characters cannot be displayed on the screen. Instead, they are recognized by some application programs as special commands.

Many of the special keys discussed before generate control-characters. For example, pressing the "I" key while holding down the "CTRL" key is equivalent to pressing the "TAB" key alone. For your reference, Table 4-1 is a list of all the special keys which generate control-characters.

Key	Control-character
TAB	CTRL-I
←	CTRL-H
→	CTRL-U
↑	CTRL-K
↓	CTRL-J
RETURN	CTRL-M
PAUSE	CTRL-S
BREAK	CTRL-C
ESC	CTRL-]
ENTER	CTRL-M

Table 4-1 Control-character keys

4.5 Function keys

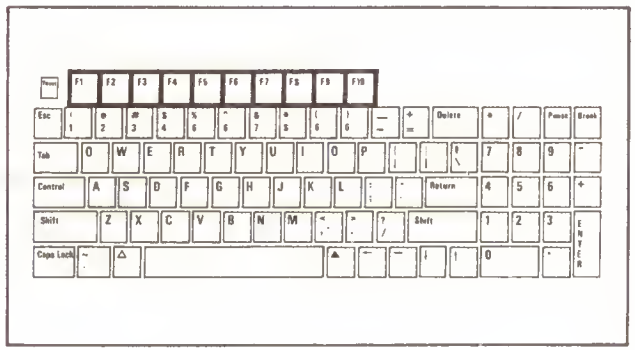


Figure 4-4 Function keys

The function keys (See Figure 4-4) at the topmost row of the keyboard provide an easy way of entering ten of the more commonly used control-characters. A control-character can be generated by pressing one of these function keys alone, without holding down the "CTRL" key. Table 4-2 is a list of the function keys and their control-character equivalents.

Key	Control-character
F1	CTRL-@
F2	CTRL-A
F3	CTRL-B
F4	CTRL-C
F5	CTRL-D
F6	CTRL-E
F7	CTRL-F
F8	CTRL-G
F9	CTRL-L
F10	CTRL-X

Table 4-2 Control-characters generated by the function keys

4.6 The numeric keypad

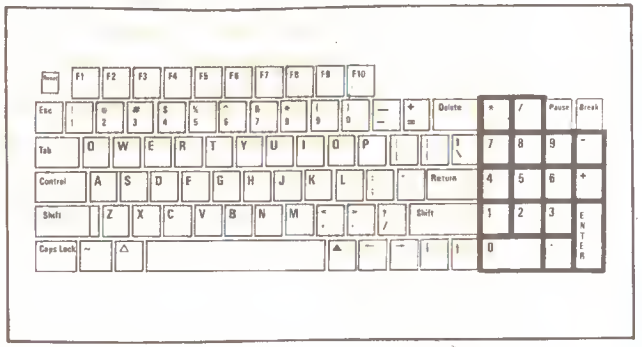


Figure 4-5 Numerie keypad

The numerie keypad (see Figure 4-5) loeated at the right side of the keyboard is just a duplicate of some of the keys on the main keypad including the number keys ("0" to "9"), the arithmetie operator keys ("+", "-", "*", "/"), the period key (".") and the "ENTER" key which is equivalent to the "RETURN" key.

The layout of these keys is as elose as possible to that of a ealeulator so that numeric data entry is made easy for a skilled operator.

CHAPTER 5

THE VIDEO DISPLAY

5. THE VIDEO DISPLAY

The video display is one of the most important output devices of a computer. It allows the user to monitor the current status of a program running on the computer and examine the computation results visually.

Besides conventional text display, the LASER 128/128EX is also capable of displaying color graphics. Graphical presentation is often more attractive than textual materials and is therefore easier to be captured.

On the other hand, text is still a simple and effective way of expressing ideas. As a compromise, the computer allows you to display both text and graphics on the screen at the same time.

In most cases, the appropriate video display mode will be selected by the application program you are using so that you don't have to worry about it. This chapter serves only as a reference for those users interested in writing programs for the computer.

5.1 Selecting a suitable video display device

Depending on the application, you may choose one of the following as the video display device:

- COMPOSITE MONOCHROME VIDEO MONITOR can produce sharp images of high-resolution graphics and text display. It is suitable for use with

software which makes extensive use of the 80-column text feature of the computer, e.g. word-processors.

- The main disadvantage is that they cannot display color which is used in many educational or game programs. However, since monochrome monitors are inexpensive, they are a popular video display device.

- COMPOSITE COLOR VIDEO MONITOR can display graphics in color, but the image is not as sharp as in monochrome monitors. Color fringes may be observed in the characters displayed in a mixed graphics/text screen.

- COLOR/BLACK-AND-WHITE TELEVISION SET is another commonly-used video display device since most people already have one. The only thing you need to buy is a TV interface which converts the composite video output of the computer to a form suitable for use with a TV.

  Color graphics can be displayed on a color TV effectively. However, since the resolution of TV is rather low, 80-column text may be difficult to be read.

- RGB COLOR MONITOR has higher resolution than composite color monitor and hence can produce high-quality color graphics and text display. However since it is more expensive, it is not quite as popular as a monitor or a TV.

- FLAT-PANEL LCD DISPLAY is another expensive display device. Like monochrome monitor, it can display high-resolution graphics and text effectively but color is not available.

  The most important advantage of LCD display over ordinary video monitors is that its size is small so that it can be carried around easily. It is particularly suitable for business applications in which portability is essential.

Please consult your dealer for further information on selecting the video display device that is most suitable for your application.

5.2 Video display modes

The computer has two text modes and four graphics modes of different resolutions. The following sections briefly discuss the various video display modes and their capabilities.

5.2.1 Text modes

The computer is capable of displaying text in either 40-column (40 characters per line) or 80-column format (80 characters per line). In both cases, 24 lines of text can be displayed on the screen at a time.

Immediately after power-up, the display is in 40-column text mode. However, if the display is somehow switched to graphics mode, you can return to text mode by typing the following BASIC command:

```
]TEXT<CR>
```

Note: <CR> means to press the RETURN or the ENTER key.

The entire screen will then display text again. The BASIC prompt character "]" and the cursor will be displayed at the bottom of the screen.

5.2.1.1 40-column text

The primary reason for having a 40-column text mode in your computer is to allow a low-resolution video monitor or TV to be used as the video display device. To switch from 80-column text to 40-column text, type the following BASIC statement:

```
]PR#0<CR>
```

The left-half of the original 80-column display is copied to the 40-column display. The BASIC prompt character "]" and the flashing checker-board cursor will be displayed on the next line as usual.

5.2.1.2 80-column text

80-column text is especially suitable for word-processing applications since the document is displayed on the screen in the same way they will be printed on paper so that you can edit the format of the print-out on the screen easily. To switch from 40-column mode to 80-column mode, type the following BASIC statement:

```
]PR#3<CR>
```

The entire screen will be cleared leaving the BASIC prompt "]" and the cursor on the top-most row of the screen. Note that the cursor becomes a non-flashing white block to inform you that you have activated the built-in 80-column firmware. Also, the width of the characters is reduced to one-half of that in 40-column text mode.

If for some reasons 80-column text is undesirable, e.g. a low-resolution TV is being used as the video display device, you can inhibit this feature by throwing the "40/80 COLUMN" switch located above the keyboard to the "40" position. If the switch is thrown to the "80" position, both 40-column and 80-column text can be displayed.

5.2.2 Graphics modes

5.2.2.1 Low-resolution graphics

The low-resolution graphics (abbreviated as lo-res) display is made up of a 40H x 48V matrix. Each picture element of the matrix, called a pixel, can take on any one of the following 16 colors shown in Table 5-1:

Code	Color	Code	Color
0	black	8	brown
1	dark red	9	orange
2	dark blue	10	gray 2
3	violet	11	pink
4	dark green	12	medium green
5	gray 1	13	yellow
6	medium blue	14	aquamariine
7	light blue	15	white

Table 5-1 Lo-res, med-res and double-hi-res colors

Two lo-res pictures can be stored in memory but you can only display one of them at a time.

5.2.2.2 Medium-resolution graphics

The medium-resolution graphics (abbreviated as med-res) screen is made up of a 80H x 48V pixel-matrix. Each pixel can take on any one of the 16 colors shown in Table 5-1. Adjacent pixels on the same line of different colors may have interference with one another. Only one med-res picture can be stored in the computer and displayed at a time.

5.2.2.3 High-resolution graphics

The high-resolution graphics (abbreviated as hi-res) screen is made up of a 280H x 192V pixel-matrix. A total of six colors can be displayed as shown below in table 5-2:

Code	Color
0	black
1	green
2	violet
3	white
4	black
5	orange
6	blue
7	white

Table 5-2 Hi-res colors

The color of a dot depends on its horizontal plotting position as well as the state of adjacent pixels:

Dots plotted on even columns of the screen will be either violet or green while dots on odd columns of the screen will be either blue or orange.

Two color dots must be separated by at least one black dot.

Two adjacent pixels which are turned on give rise to a continuous white dot.

As a result, the effective "color-resolution" of hi-res mode is 140H x 192V. Like lo-res graphics, two hi-res pictures can be stored in the computer at the same time but only one of them can be displayed at a time.

5.2.2.4 Double-high-resolution graphics

The double-high-resolution graphics (abbreviated as double-Hi-res) screen is divided into 560H x 192V pixels. Only one display page is available as in med-res graphics. Depending on the horizontal plotting position of a dot and the state of its adjacent pixels, one of the 16 colors shown in table 5-1 can be displayed.

Adjacent color dots may interfere with one another. The exact mechanism of the double-hi-res color generation scheme is quite complicated and will not be detailed here. For more information, refer to the *Technical Reference Manual.*

5.2.3 Mixed graphics/text display

For any of the four graphics modes described in section 5.2.2, there is a mixed graphics/text mode available. A mixed mode screen is made up of a graphics display on the upper portion and four lines of text at the bottom as shown in Figure 5-3.

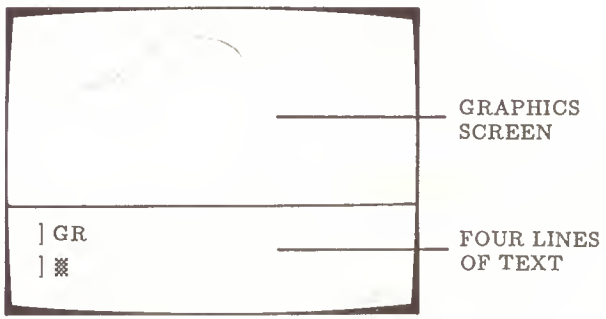


Figure 5-3 Mixed graphics/text display

For lo-res and hi-res graphics, the four lines of text can be displayed in either 40-column or 80-column format. For med-res and double-hi-res graphics, the text is always displayed in 80-column format.

For example, to select mixed lo-res graphics/text display, type the following BASIC statement:

```
]GR<CR>
```

The upper portion of the screen will be switched to lo-res graphics mode and cleared while the BASIC prompt character "]" and the cursor will be displayed right below the lo-res graphics screen. Refer PART II of this manual for more information on programming the lo-res graphics display in BASIC.

CHAPTER 6

INPUT/OUTPUT PORTS

6. INPUT/OUTPUT PORTS

A computer on its own is just a calculating machine. To be useful, it must be able to communicate with the real world by some means. This is the job of the Input/Output (abbreviated as I/O) ports. In the LASER 128/128EX computer system, there are eight I/O ports which correspond to the peripheral slots in an Apple® IIe computer.

All the I/O ports have already been occupied by built-in I/O devices and interfaces for commonly used I/O devices so that in most cases you do not need to purchase any other peripheral interface cards.

However, if you need to install additional I/O devices for some special applications, there is still a way to do it. On the left side of your computer there is an expansion connector for plugging in an optional FCC approved expansion box in which up to two peripheral interface cards can be installed. This opens up your computer system to almost all kinds of applications.

6.1 Built-in I/O devices and interfaces

A port number is assigned to each of the built-in I/O devices and interfaces as shown in table 6-1.

Port number	I/O device or interface
0	Built-in 40-column display
1	Parallel/Serial printer interface
2	Serial communication interface
3	Built-in 80-column display
4	Mouse interface
5	1 Megbytc expansion RAM interface
6	Built-in 5.25" disk drive interface
7	3.5" disk drive interface

Table 6-1 I/O port assignment

6.1.1 Port 0 - 40-column display

The 40-column text display capability of the computer has already been described in section 5.2.1.1. To enter 40-column text mode in BASIC, type the following statement:

```
]PR#0<CR>
```

When the computer is in 40-column text mode (indicated by a flashing checker-board cursor), the following control-characters will be recognized as special commands:

CTRL-G: "Beeps" the speaker.

CTRL-H: Moves the cursor left one character position. Functionally equivalent to the "←" key.

CTRL-J: Moves the cursor down one line. Functionally equivalent to the "↓".

CTRL-M: Moves the cursor to the leftmost position of the following line. Functionally equivalent to the "RETURN" or "ENTER" keys.

In addition to the above control-character commands, the computer also recognize some two-character commands in the 40-column text mode. The command sequence is made of a "lead-in" character ("ESC") followed by the actual command character. The following is a list of all the available "ESCape-code" commands:

ESC @: Clears the screen and places the cursor at the upper-left-hand corner of the screen.

ESC E: Clears the display from the current cursor position to the end of the line.

ESC F: Clears the display from the current cursor position to the bottom of the screen.

After pressing the "ESC" key, the following "ESCape-code" commands can be entered repeatedly by pressing the command key alone:

ESC I: Moves the cursor up one line.

ESC J: Moves the cursor left one character position.

ESC K: Moves the cursor right one character position.

ESC M: Moves the cursor down one line.

For example, pressing the "ESC" key once and "I" key twice moves the cursor up by two lines.

6.1.2 Port 1 - Parallel/Serial printer

On the back panel of the computer, there is a parallel printer connector for connecting a Centronics type parallel printer and a serial printer connector for connecting a serial printer.

The "Parallel/Serial" switch located above the keyboard selects either the parallel printer or the serial printer as the hardcopy-printing device. To activate the printer port, type the following statement while you are in BASIC command level:

]PR#1<CR>

The following characters you type will then be printed on the printer whenever the "RETURN" key is pressed.

6.1.2.1 Parallel printer commands

When the parallel printer is activated, the computer will recognize several parallel printer commands. The printer command is made up of a "lead-in" character (CTRL-I) and a command character code. The following is a list of the available parallel printer commands:

CTRL-I "c": Change the printer command "lead-in" character from CTRL-I to the control-character "c". For example, typing "CTRL-I CTRL-P" changes the "lead-in" character to "CTRL-P". Subsequent printer commands must begin with a "CTRL-P", followed by the actual command character.

CTRL-I X: Don't send all eight bits of each character code to the printer. This is the usual setting.

CTRL-I H: Send all eight bits of each character code to the printer. This is particularly useful for printing graphics.

CTRL-I I: Echo the text being printed back to the screen. If the line-width of the printer is greater than that of the current text display (either 40-column or 80-column), then echoing may cause problem with the print-out.

CTRL-I nnnN: Turn off screen-echoing and set the printer line-width to "nnn", where "nnn" is a decimal number from 0 to 255. For example, CTRL-I 80N sets the line-width to 80 column. CTRL-I 0N sets "no" line-width.

CTRL-I L: Automatically print a linefeed character after each carriage-return.

CTRL-I K: Do not print a linefeed character after each carriage-return automatically.

CTRL-I Z: "ZAP" mode: Don't check for any more commands.

CTRL-I c: Insert carriage returns whenever the line width (set by CTRL-I nnnN) is exceeded.

6.1.2.2 Serial printer commands

When the serial printer is activated, the computer will recognize the following serial printer commands:

CTRL-I "c": Change the printer command "lead-in" character from CTRL-I to the control-character "c".

CTRL-I I: Echo the text being printed back to the screen. If the line-width of the printer is greater than that of the current text display (either 40-column or 80-column), then echoing may cause problem with the print-out.

CTRL-I nnnN: Turn off screen-echoing and set the printer line-width to "nnn", where "nnn" is a decimal number from 0 to 255. For example, CTRL-I 80N sets the line-width to 80 column. CTRL-I 0N sets "no" line-width.

CTRL-I L: Automatically print a linefeed character after each carriage return.

CTRL-I K: Do not print a linefeed character after each carriage return automatically.

CTRL-I Z: "ZAP" mode: Don't check for any more commands.

CTRL-1 C: Insert carriage returns whenever the line width (set by CTRL-1 nnnN) is exceed.

CTRL-I nnB: Set the baud rate according to the number "nn" as in Table 6-2:

nn	Baud Rate
1	50
2	75
3	110
4	135
5	150
6	300
7	600
8	1200
9	1800
10	2400
11	3600
12	4800
13	7200
14	9600
15	19200

Table 6-2 Serial printer baud rate settings

CTRL-I nD: Set the serial data format according to the number "n" as in table 6-3:

n	Data Format
0	8 data bits, 1 stop bit
1	7 data bits, 1 stop bit
2	6 data bits, 1 stop bit
3	5 data bits, 1 stop bit
4	8 data bits, 2 stop bits
5	7 data bits, 2 stop bits
6	6 data bits, 2 stop bits
7	5 data bits, 2 stop bits

Table 6-3 Serial printer data formats

CTRL-I nP: Set the parity according to the number "n" as follows:

n	Parity
0	none
1	odd
3	even
5	mark
7	space

Table 6-4 Serial printer data parity settings

6.1.2.3 PRINTING GRAPHICS WITH THE COMPUTER

Many programs such as Printshop, Newsroom, Dazzle Draw, Mouse Paint, and Printographer allow you to print graphics on various printers. You can use them with your computer. They only thing you need to know is how to set up the software so it knows how to talk to the computer's parallel and serial printer ports. to make this easier, here is printer setup information for several popular graphics printing programs. If one you use isn't on this list, you will need to experiment with various options until you find one that works.

If you have a parallel printer:

Select the Apple Parallel Card or the Epson APL Card (except when using some parallel printers, e.g. Star printer, with Printshop, in which case you should select the Super Serial Card interface instead)

If you have a serial printer:

Printshop:	Select the Super Serial interface card
Printographer:	Select Super Serial or IIc interface card

Newsroom: Select the "Firmware 1.0 protocol" and select the "ZAP" mode using the computer Port configuration program (CTRL-P-RESET)

Mousepaint: Mouse Paint will only recognize an image-writer printer, which must be set to do a line-feed after carriage retrun (This must be done on the printer - Port configuration program).

6.1.3 Port 2 - Serial communication

On the back panel of your computer, you can find a connector similar to the serial printer connector for connecting serial communication devices such as modems. To activate the serial port in BASIC command level, type the following statement:

]PR#2<CR>

The subsequent characters you type will then be sent to the communication device which is connected to the serial interface connector.

When the serial port is aetivated, the computer will reeognize several eommunieation eommands. A communieation eommand is made up of a "lead-in" eharacter ("CTRL-A") and a eommand eharaeter. The following is a list of all the serial eommunieation eommands available:

CTRL-A "e": Change the eommunication eommand "lead-in" character from "CTRL-A" to the control-eharaeter "e".

CTRL-A I: Eeho the eharacters sent to the serial port baek to thc sereen.

CTRL-A L: Automatieally print a linefeed character after eaeh carriage-return.

CTRL-A K: Do not print a linefeed eharaeter after eaeh earriage-rcturn automatieally.

CTRL-A Z: "ZAP" mode: Do not eheek for any more eommands.

CTRL-A nnB: Set the baud rate according to the number "nn" as in Table 6-5:

nn	Baud Rate
1	50
2	75
3	110
4	135
5	150
6	300
7	600
8	1200
9	1800
10	2400
11	3600
12	4800
13	7200
14	9600
15	19200

Table 6-5 Serial port baud rate settings

CTRL-A nD: Set the data format according to the number "n" as in Table 6-6:

n	Data Format
0	8 data bits, 1 stop bit
1	7 data bits, 1 stop bit
2	6 data bits, 1 stop bit
3	5 data bits, 1 stop bit
4	8 data bits, 2 stop bits
5	7 data bits, 2 stop bits
6	6 data bits, 2 stop bits
7	5 data bits, 2 stop bits

Table 6-6 Serial communication data formats

CTRL-I nP: Set the parity according to the number "n" as in table 6-7:

n	Parity
0	no
1	odd
3	even
5	mark
7	space

Table 6-7 Serial communication data parity settings

6.1.4 Port 3 - 80-column display

The LASER 128/128EX normally displays text in 40-column format. To switch to 80-column mode, type the following BASIC statement:

]PR#3<CR>

When the computer is in 80-column text mode (indicated by a non-flashing solid cursor), the following control-characters are recognized as special commands:

CTRL-G : "Beeps" the speaker.

CTRL-H : Moves the cursor left one character position. Functionally equivalent to the "←" key.

CTRl-J : Moves the cursor down one line. Functionally equivalent to the "↓" key.

CTRL-M : Moves the cursor to the leftmost position of the following line. Functionally equivalent to the "RETURN" and "ENTER" keys.

Notice that the above commands used are also valid in 40-column mode.

The following "control-character" commands will only be recognized if the 80-column firmware is activated:

CTRL-E : If Apple® Pascal is being used, it enables the visible cursor, displaying it as each character is being printed on the screen. This is the usual setting for Pascal.

CTRL-F : If Apple® Pascal is being used, it disables the visible cursor. Text display is faster if the visible cursor is disabled.

CTRL-K : Clears the display from the current cursor position to the bottom of the screen.

CTRL-L : Clears the screen and moves the cursor to the upper-left-hand corner of the screen.

CTRL-N : Displays subsequent text in "normal" format, i.e. white characters on a black background.

CTRL-O : Displays subsequent text in "inverse" format, i.e. black characters on a white background.

CTRL-Q : Switches back to 40-column display while keeping 80-column features. Notice that the cursor will still be a non-flashing white block as in 80-column mode instead of the ordinary flashing checker-board pattern.

CTRL-R : Returns to 80-column display.

CTRL-U: Switches back to 40-column display and disables 80-column features. A flashing checker-board cursor will be displayed instead of the non-flashing solid cursor in 80-column text mode.

CTRL-W : Scrolls the screen up one line without moving the cursor.

CTRL-X : Do not display special graphics characters.

CTRL-Y : Moves the cursor to the upper-left-hand corner of the display without clearing the screen.

CTRL-Z : Clears the entire line the cursor position on.

CTRL-I : Displays special graphics characters if inverse text mode is also selected.

CTRL-\ : Moves the cursor one position to the right.

CTRL-] : Clears from the current cursor position to the end of the line.

CTRL-_ : Moves the cursor up one line.

Besides the above control-character commands, some "ESCape-code" commands are also recognized by the computer in 80-column text mode. They are two-character commands made up of a "lead-in" character ("ESC") and a command character. The available "ESCape-code" commands are listed below:

ESC @ : Clears the screen and places the cursor at the upper-left-hand corner of the screen.

ESC E : Clears the display from the current cursor position to the end of the line.

ESC F : Clears the display from the current cursor position to the bottom of the screen.

ESC 4 : Switches to 40-column display while keeping 80-column features.

ESC 8 : Returns to 80-column display.

ESC CTRL-Q : Switches back to 40-column display and disables 80-column features.

ESC CTRL-D : Disables recognition of control-character commands which apply to 80-column mode only.

ESC CTRL-E: Enables recognition of extra 80-column control-character commands.

After pressing the "ESC" key, the following "ESCape-code" commands can be entered repeatedly by pressing the command key alone:

ESC I : Moves the cursor up one line.

ESC J : Moves the cursor left one character position.

ESC K : Moves the cursor right one character position.

ESC M : Moves the cursor down one line.

6.1.5 Port 4 - Mouse

A mouse can be connected to the 9-pin game connector located on the back panel of the computer. Usually, the application programs which make use of the mouse as an input device will manipulate it automatically so that you don't have to worry about it.

However if you are interested in writing programs for the mouse yourself, you can refer to the *Technical Reference Manual* for a detailed description of the operation of the mouse.

6.1.6 Port 5 - Expansion memory

The computer comes with 128K system RAM. This is more than sufficient for most applications. However, if it still cannot satisfy your needs, you can easily expand the RAM size up to 1M-bytes!

1M-bytes=1024 K-bytes
=1,048,576 bytes

This expansion memory is treated as one of the I/O devices of the computer which is connected to port 5.

To install additional RAM in the computer, please refer to APPENDIX B.

Unlike the system RAM, the expansion RAM is not directly addressable and hence cannot be used as program memory, i.e. a program cannot run on it. Instead, it is primarily designed for temporary data storage. A typical application of the expansion RAM is to use it as a "RAM disk", i.e. emulating a very high-speed floppy disk system for temporary storage of application programs or data files.

Before you begin to work with the programs and data files stored on a floppy disk, you may load the files, or even the contents of the entire diskette (if the size of the expansion memory is large enough), into the expansion RAM and work with this "RAM disk" instead of the actual diskette. Since no mechanical movement is involved in the operation of the "RAM disk", the information can be accessed much faster than a real floppy disk system.

After you have finished with your work, you can then store the contents of the "RAM disk", which may be modified, back to the actual floppy disk. Notice that since the floppy disk is only accessed twice in the entire process, damage to the diskette due to mechanical movement is greatly reduced.

The expansion RAM is recognized by some operating systems (e.g. Pascal version 1.3 or later versions and ProDOS®) and will be formatted as a storage device automatically. However if you want to use the expansion RAM directly, you can refer to the *Technical Reference Manual* for a detailed description of the operation of the expansion RAM.

6.1.7 Port 6 - 5.25" disk drive

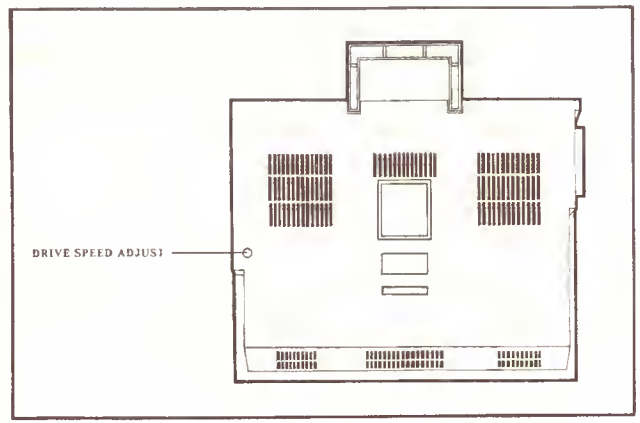
The computer has a built-in 5.25" disk drive and an external drive connector located on the back panel for connecting an external drive.

The built-in disk drive is assigned as drive 1 of port 6. Any 5.25" drive connected to the external drive connector is assigned as port 6 drive 2, and any 3.5" drives plugged into the external drive connector are assigned as port 7 drives 1 and 2. To activate, the built-in disk drive, type the following Basic statement.

```
]PR#6<CR>
```

You will then hear some clattering noises as the internal disk drive pulls back the disk arm to the outermost track. If an auto-startup program diskette is inserted in the drive and the drive door is closed, then the program will be loaded and run automatically. If you have a 3.5" drive connected to the external drive connector and the drive contains an "auto-startup" disk, the "booting" process will start from this drive. Otherwise, the "booting" process will normally start from the built-in disk drive. Refer to the *Technical Reference Manual* for a more detailed description of the drive port and its operation. The external drive will be handled by the application program that you are using.

6.1.7.1 ADJUSTING DRIVE SPEED



The Drive Speed Adjust, also on the bottom of the unit, allows you to adjust the speed of the internal disk drive with a small screw-driver. This is included so that certain drive speed critical application programs can be run successfully. You may also need to adjust the drive when the computer is operating at extreme temperature. Some software packages include utility to help you to check drive speed. E.g. Locksmith or Filer etc.

Warning: In general, you need not adjust the drive speed.

6.1.8 Port 7 - 3.5" disk drive

The 3.5" disk drive at the external drive connector can be accessed at port 7 of the computer. To turn on drive 1 of port 7, type the following BASIC statement:

```
]PR#7<CR>
```

The program stored on the diskette inserted in the drive will be loaded and run automatically as described before.

6.2 Expansion connector

On the left side of the LASER 128/128EX is a 50-pin connector for plugging in a separately purchased expansion box which can support two peripheral cards.

Besides providing mechanical protection to the interface cards, the expansion box, with its own power supply, also helps to relieve the loading on the power supply of the computer.

One of the two 50-pin connectors in the optional expansion box is for installing external I/O devices in port 7 of the computer.

The "INT/EXT PORT 7" switch inside the ROM door selects either the built-in 3.5" disk drive interface or the peripheral card plugged into the expansion box as the I/O device at port 7.

When the switch is thrown to the "INT PORT 7" position, the built-in 3.5" drive interface is selected. Otherwise, the peripheral card plugged into the expansion box is selected.

The other connector emulates slot 5 of an Apple® IIIe computer. The "INT/EXT PORT 5" switch located by the side of the "INT/EXT PORT 7" switch selects either the built-in expansion RAM interface or the external interface card plugged into this connector as the I/O device at port 5 of the computer.

For more information on installation of external interface cards, refer to the user's manual that comes with the expansion box.

CAUTION: Turn all power off prior to connecting or disconnecting peripherals.

(This page is intentionally left blank)

PART II

BASIC MANUAL

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The computer's BASIC is a full implementation of the most popular microcomputer programming language. BASIC is run through a program called an Interpreter.

This allows you to enter BASIC commands and have them executed immediately.

1.1 TO START BASIC

Turn on your computer. The logo and the BASIC prompt sign "]" will appear on your display monitor. Immediately beside the prompt sign there will be a flashing cursor.

For the preparations before power up, please refer to the User's Manual.

The machine is now ready for you to enter BASIC commands, or BASIC program.

1.2 THIS MANUAL

The manual is divided into four sections:

- Some background information on BASIC programming - this chapter describes how the computer handles its implementation of BASIC, as well as giving you general information on the strengths and limitations of the language.

- Computer's BASIC Statements - this chapter presents all the statements and commands used in BASIC. It is arranged in alphabetical order.

- Computer's BASIC Functions - this presents all of the computer's built-in BASIC functions, and it is also arranged alphabetically.

- Appendices - these contain the ASCII and keyboard character codes, error messages, and a list of reserved words.

This BASIC manual completely describes the language as it is implemented on the computer.

However, it is not intended as a guide to learning the language, although if you followed through the descriptions of the commands and functions a number of times you would get a fair grasp of it.

If you are completely new to BASIC, there are a large number of books available to help you learn it. Among them are:

BASIC from the ground up, by David E. Simon, Hayden, 1978,

BASIC, by Robert L. Albrecht, Leroy Finkel, and Jerry Brown, John Wiley & Sons, 1973.

1.3 VARIABLES

Variables are the names you assign to values that change in your BASIC program.

The values can be given directly - initialised - as in this example:

A = 63.999

, or they can take up new values as a result of the program's execution. For instance, in the next example, the value of I varies from 1 to 10.

```
] 10 FOR I = 1 TO 10
] 20 PRINT I
] 30 NEXT
```

When a BASIC program starts running, all variables that have no explicitly assigned values (as in the first example) are assumed to be zero.

1.4 VARIABLE NAMES

BASIC variable names must start with an alphabetic letter. They can be up to 40 characters long, and can represent either numbers or strings.

Strings are groups of letters and symbols.

The variable names cannot be reserved words - for a list of reserved words see the Appendices - nor can they have reserved words embedded in them. For instance **ADIMMY** contains the reserved word **DIM**, and is not an allowable variable; and **DIMMY** starts with the reserved word, and is not permissible.

Reserved words include all the BASIC commands, statements, functions, and operator names.

Integer variables are denoted by a percentage sign "%" immediately following the variable names. This type of variable can only contain integer values, for example:

```
I% = 1234
```

String variables must always end with a dollar sign "$". This declares to the BASIC interpreter that it is dealing with string variables, and it will allocate extra memory to handle it. For example:

```
SENTENCE$ = "MY FIRST STRING VARIABLE!"
```

SENTENCE$ is a valid string variable, but SENTENCE - without the $ sign at the end - is not.

1.5 ARRAY VARIABLES

An array is a matrix or table of values that is referenced by the same variable name. The specific values are accessed by subscripts which are used in conjunction with the array's name.

The number of subscripts for an array is the same as the number of dimensions for it, and are defined in the DIM statement. For instance:

DIM ARRAY (10, 10, 10) sets up a three-dimensional array where the subscript for each dimension ranges from 0 to 10. Thus it is equivalent to a table containing 11 x 11 x 11 or 1331 entries.

DIM BARRAY (9) sets up a one-dimensional array with a single subscript which can range from 0 to 9. Thus it is equivalent to a table of 10 entries.

1.6 EXPRESSIONS AND OPERATORS

An expression can be a constant, or a numeric variable, or a string, or a combination of variables, constants, and operators which work together to produce a single value.

There are four types of operators:

- arithmetical
- logical
- relational
- functional

1.6.1 ARITHMETICAL OPERATORS

These are the common mathematical operators, and they are always executed in a set order of preference. They are listed below in this order:

Operator	Operation	Example
^	Exponentiation	A ^ B
-	Negation	-A
*, /	Multiplication, division	A * B
+, -	Addition, subtraction	A - B

This order of operations can be changed by using parentheses, as the expressions within parentheses are evaluated first.

Within parentheses, the above order is kept. Following are some examples of how this is done:

Algebraic expression	BASIC expression	Result
2 + 10 ÷ 2	2 + 10/2	7
(2 + 10) ÷ 2	(2 + 10)/2	6

Operators of the same preference are executed form left to right.

1.6.2 LOGICAL OPERATORS

These operators work on values according to their logical states to produce a result which is either one "1" or zero "0" - "true" or "false". A non-zero value corresponds to a "true" state, while a zero value corresponds to a "false" state. The outcome of a logical operation is as shown in the table following. The operators are listed in order of precedence.

NOT

A	NOT A
1	0
0	1

AND

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

OR

A	B	A OR B
1	1	1
1	0	1
0	1	1
0	0	0

1.6.3 RELATIONAL OPERATORS

These operators share some similarities with the logical operators, in that their result can only be a zero or a one ("false" or "true").

Like the logical operators, these relational operators can be used to make decisions regarding program branching.

Operator	Relation	Example
=	Equality	A = B
< >	Inequality	A < > B
<	Less than	A < B
>	Greater than	A > B
< =	Less than or equal to	A < = B
> =	Greater than or equal to	A > = B

The equal sign "=" is also used to assign a value to a variable (see LET statement in Chapter 3 of this manual.)

If arithmetical, relational, and logical operators are combined in one expression, the order of precedence of evaluation is: arithmetic, then relational, then logical.

1.6.4 FUNCTIONAL OPERATORS

BASIC has a number of built-in or "intrinsic" functions which may be used in either direct or indirect mode. These are described in Chapter 4.

Examples are TAN(A), which calculates the tangent of an angle A, and LEFT$(X$, 3), which returns the three leftmost characters of a string X$.

You can define your own functions using the DEF FN command. (see DEF FN command in Chapter 3)

1.7 STRING OPERATIONS

Two strings can be compared using the relational operators. They work by comparing the numeric ASCII values of each corresponding character of each string.

The conditions for equality or inequality depend on whether the ASCII codes are higher or lower.

Example:

```
] 10 IF "LASER" > "RESAL" GOTO 30
] 20 PRINT "LASER"
] 30 END
RUN
LASER
```

Two strings can be combined - concatenated - using the plus "+" operator.

Example:

```
] 50 X$ = "COMPUTER  "
] 60 Y$ = "OK"
] 70 PRINT X$ + Y$
RUN
COMPUTER OK
```


CHAPTER 2

SOME BACKGROUND IN BASIC PROGRAMMING

2 SOME BACKGROUND IN BASIC PROGRAMMING

When BASIC is started, it displays the prompt "]". This means that it is ready to accept commands from the keyboard.

At this command level, it can be used in either of two modes:
Direct - which is when you enter a command and have it executed immediately.
Indirect - which is when command lines are started with line numbers, and a program is built up for later execution.

2.1 LINE FORMAT

In the direct mode, the BASIC commands and functions are entered as they are.

In the indirect mode, program lines like the one following are entered:

nnnnn *BASIC statement (: BASIC statement . . .)* Where nnnnn is the line number.

The parentheses indicate options. The length of your line is limited to 239 characters, and a line input is terminated when you hit the RETURN key.

Hitting RETURN adds a non-printing carriage return character at the end of a line. The BASIC interpreter takes this carriage return as indicating the end of a program line.

The line numbers must be in the range of 0 to 63999. They relate to the order in which a BASIC program is stored in memory, and the interpreter always executes a program in the sequence of the line numbers (unless the program branches otherwise.

2.2 CONSTANTS

As their name implies, thes are values that do not change. In BASIC they can be either numeric or string value. Some string constants are:

"$64,000"

"May the Force be with you'

There are two types of numeric constant.

1. Integer constants
   Whole numbers in the range from -32767 to +32767.

2. Floating point constants
   Positive or negative numbers that are represented in exponenial form. These are made up of three parts: the fixed point part, in decimal form; the E which signifies exponentiaton and the exponent, which must be an integer. The range of values for floating point constants is from 1E-38 to 8.5E + 37.

   Example:

```
256.1024E-7 = .0000256124
4096E7= 40960000000
```

2.3 USING A PRINTER WITH THE COMPUTER

The computer has two built-in printer interfaces : a serial printer interface and a parallel printer interface. To select either of them, put the PARALLEL/SERIAL switch in the proper position.

2.4 SERIAL PRINTER

The serial interface is RS232 standard and may be connected to any printer with this interface standard.

To activate the serial printer, follow these steps:

1. Connect the interface cable between the computer and the serial printer.

2. Set up your printer according to the following requirements:

   i. Baud-rate: this is the rate of data transfer between your computer and printer so they must match each other. Set the desired baud-rate for your printer first and remember the number. You will have to set up your computer later using this number.

   ii. No of Bits : 8

   iii. Parity : No

   iv. Stop Bits : 1

3. Set up your computer by pressing "CTRL", "RESET" and "P" all at the same time.

   Following is a table of all the options of the serial printer configuration you will see on screen:

	Bit	Baud	Prty	Echo	LF	Wdth	CR	Zap
	6	6	1	1	2	4	1	1
1	6/1	110	NONE	NO	NO	NONE	NO	NO
2	6/2	300	EVEN	YES	YES	40	YES	YES
3	7/1	1200	ODD			72		
4	7/2	2400	MARK			80		
5	8/1	4800	SPACE			132		
6	8/2	9600						
7		19200						

4. Initialise the serial printer by typing on the keyboard.
   PR # 1 if you are in BASIC
   1 CTRL-P if you are in Monitor

   (1 CTRL-P means you type a "1", then, while holding down the "CTRL" key, press "P")

5. From now on, any character displayed on the screen will be echoed to the printer.

6. To stop printing, type
   PR # 0 if you are in BASIC
   0 CTRL-P if you are in Monitor

2.5 PARALLEL PRINTER

The parallel printer interface is Centronics standard.

To activated the parallel printer, follow these steps:

1. Connect the interface cable between the computer and the parallel printer.

2. Set up your printer following the printer's manual. Usually you will place it in the ON-LINE mode.

3. Initialise the parallel printer by typing
   PR # 1 if you are in BASIC
   1 CTRL-P if you are in Monitor

4. From now on any character displayed on the screen will be echoed to the printer.

5. To stop printing, type
   PR # 0 if you are in BASIC
   0 CTRL-P if you are in Monitor

CHAPTER 3

BASIC COMMANDS AND STATEMENTS

3 BASIC COMMANDS AND STATEMENTS

All of the computer's BASIC commands and statements are given in this chapter, with each laid out as follows:

Purpose: Tells what the command or statement is used for.

Format: Shows the correct layout for the command or statement. You will be able to follow the layout if you keep the following rules in mind:

1. Words given in capital letters must be input exactly as shown.
2. You must enter any item given in lower case italic letters.
3. Items indicated in square brackets are optional [optional].
4. Items followed by three periods . . . mean that the particular item may be repeated as often as you like.
5. Quotation marks, commas, full-stops, hyphens, semicolons, and equal signs must be used as indicated.

Comments: Describes the circumstances in which the command is used.

Example: Gives sample programs or program sections in which the command or statement is used.

3.1 AMPERSAND COMMAND(&)

Purpose: To jump into a machine language command starting at hex location $3F5.

Format: &

Comments: A machine language subroutine must be placed at $3F5 before using this command, otherwise an unexpected result may occur, which may even destroy your program.

Example:

```
] CALL-151
* 3F5 : 4C 00 C3
* CTRL-C
] &
]
```

You enter the system monitor program and place a JUMP machine instruction at location $3F5. Executing the AMPERSAND COMMAND will direct control to address $C300 where the 80 column display firmware is located.

3.2 CALL

Purpose: To use an assembly language subroutine.

Format: CALL *expression*

Comments: This statement transfers program flow to an assembly language subroutine.

expression is the entry address of the machine language routine and it must be in decimal.

Example:
```
] 300 ASSEM = 64600
] 310 CALL ASSEM
] RUN
```

This CALL returns control to the ROM-based monitor program, which then clears the screen, and displays the prompt in the HOME position.

3.3 CHR$

Purpose: Converts an ASCII code to its equivalent character.

Format: CHR$ *(n)*

Comments: This operation returns the single character corresponding to the number *n*, which must be between 0 and 255.

The ASCII character codes are listed in Appendix G.

Example: The following example would print all the uppercase letters of the alphabet (ASCII codes 65 through 90).

```
] 10 FOR I = 65 TO 90
] 20 PRINT CHR$(I);
] 30 NEXT I
```

3.4 CLEAR

Purpose: To clear all variables, arrays and strings.

Format: **CLEAR**

Comments: All numeric variables will be cleared to zero.
All string variables will be cleared to "null-string".

Example:

```
] 10 A = 10 : B$ = "GOOD STRING"
] 20 PRINT A, B$
] 30 CLEAR
] 40 PRINT A, B$
] RUN
10   GOOD STRING
0
```


3.5 COLOR

Purpose: To set the color of subsequently plotted low resolution graphics.

Format: **COLOR** = *color code*

Comments: For low resolution graphics, we have two sets of colors depending on the type of monitor you are using. With a composite monitor, the colors given by color codes are as follows:

Code	Color
0	black
1	magenta
2	dark blue
3	purple
4	dark green
5	grey1
6	medium blue
7	light blue
8	brown
9	orange
10	grey2
11	pink
12	light green
13	yellow
14	aquamarine
15	white

However, with a RGB monitor, the colors are slightly different. They are given as follows:

Code	Color
0	black
1	magenta
2	dark blue
3	purple
4	dark green
5	dark yellow
6	medium blue
7	grey
8	black
9	orange
10	dark blue
11	pink
12	light green
13	yellow
14	aquamarine
15	white

Example:

```
] 10 GR
] 20 FOR I = 0 TO 15
] 30 COLOR = I
] 40 VLIN 0, 39 AT I
] 50 NEXT
```

Running this program, the 16 colors will be displayed in vertical bars.

3.6 CONT

Purpose: To restart a program again after it has been halted.

Format: **CONT**

Comments: The program resumes at the next instruction after the break occurred.

CONT is often used in debugging a program, in conjunction with STOP. Once the program has been halted, you can examine and change variables in the program, and then use CONT to resume it. CONT may not work if you modify the program while it is halted.

Example: See CONT used in the example for the STOP statement.

3.7 DATA

Purpose: To store constant numbers and string values in your porgram so they can be used in conjunction with the READ statement.

Format: **DATA** *constant (, constant) . . .*

Comments: DATA statements are non-executable and may be placed anywhere in the program.

No numeric or string expressions can be used in the DATA statement. The constant may be a number or a string. There is no need to enclose a string with quotation marks ("), but any spaces in between are ignored.

The numeric constants may be in any format, i.e., fixed point, floating point or integer.

The variable type given in the READ statement must agree with the corresponding constant in the DATA statement.

Example: See examples for the READ statement.

3.8 DEF FN

Purpose: To define and name a functiom that is written by the user.

Format: **DEF FN** *name (real variable)* = *expression.*

Comments: The *name* is exactly the same as a variable name. A user-defined string function is not allowed. The *real variable* is the variable that will be used when the function is evaluated.

The expression can be as long as a line (239 characters long).

If you need to program functions that require more room than that, you should implement your function as a subroutine.

Example:

```
] 10 GEE = 9.8
] 20 DEF FN DIS(T)= GEE * T
  ^ 2/2
] 30 INPUT "Time?";T
] 40 PRINT "Distance is";FN
  DIS(T)
```

This would calculate the distance that a body has fallem after T seconds, using the function DIS which is derived from the formula $s=1/2gt^2$, where s is the distance, g the acceleration due to gravity, and t the time that has elapsed since the object was dropped from a stationary position.

3.9 DEL

Purpose: Removes program lines.

Format: **DEL** *line number 1, line number 2*

Comments: This deletes the lines from *line number 1* to *line number 2*, inclusively.

Example: DEL 10, 110

This removes all the lines between 10 and 110, including lines 10 and 110.

3.10 DIM

Purpose: This sets the maximum subscripts for a variable and allocates enough storage to accomodate them.

Format: **DIM** *variable (subscripts)* [, *variable (subscripts), . . .*]

Comments: When the BASIC interpreter encounters a DIM statement, it initialises all the elments of the array to zero, if it is a numeric array.

For a string array, all elements are initially null strings (i.e. empty strings). However the length of each element can be different as a result of program execution.

If an array is used in a BASIC program without a corresponding DIM statement, the interpreter assumes the value of the subscript to be 10.

The maximum number of dimensions and maximum number of elements in each dimension depend on the amount of free memory in the system.

Example:

```
] 10 DIM A(10, 10)
] 20 FOR I = 1 TO 10
] 30 FOR J = 1 TO 10
] 40 IF I = J THEN A (I, J) = 1
] 50 PRINT A (I, J) = 1
] 60 NEXT J
] 70 PRINT
] 80 NEXT I
] 90 END
```

This will build up an array whose diagonal elements are all ones, with the rest of the elements remaining zero.

3.11 DRAW

Purpose: To draw geometric shapes

Format: **DRAW** *shape (shape parameters,. . .)*

Comments:

3.11.1 Drawing shapes

In the high resolution graphics modes, you can draw and move around free-form shapes.

The general graphics commands of the computer HPLOT and PLOT, only give static shapes. With shapes you defined, you can animate your creations, either moving, rotating, or changing their sizes.

3.11.2 Setting up the shapes

The first step is to sketch on paper the shape you want, and then break this down into a series of directed lines (ie. vectors). For instance, a rectangle could be broken down like this:

Figure 1. Vectors for a rectangle

To create a triangle, your shape definition will look something like:

3RD VECTOR (UNUSED)	2ND VECTOR	1ST VECTOR	HEX. DATA	COMMENT
00	101	100	2C	move up and right with plot
00	101	100	2C	move up and right with plot
00	101	100	2C	move up and right with plot
00	101	100	2C	move up and right with plot
00	101	110	2E	move down and right with plot
00	101	110	2E	move down and right with plot
00	101	110	2E	move down and right with plot
00	111	110	3E	move down and left with plot
00	111	111	3F	move left and left with plot
00	111	111	3F	move left and left with plot
00	111	111	3F	move left and left with plot

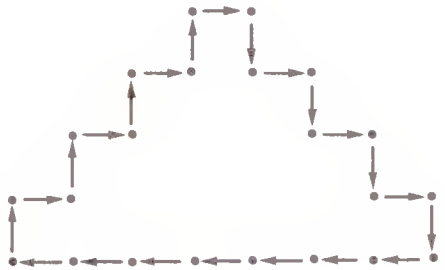


Figure 2. Vectors for a triangle

3.11.3 The shape table

In the previous pages, you have learnt how to create a single shape definition as a whole shape table.

But, in fact, a shape table can consist of more than one shape definition so that more than one shape can be manipulated by using the DRAW, XDRAW, ROT and SCALE commands.

Figure 3 shows the genral format of a shape table. You can see that the first few bytes of the shape table are used to tell the computer how many shape definitions are within the shape table, and where these shape definitions are, relative to the starting address of the shape table. The last byte of your shape definition must be zero to signify the end of the shape table.

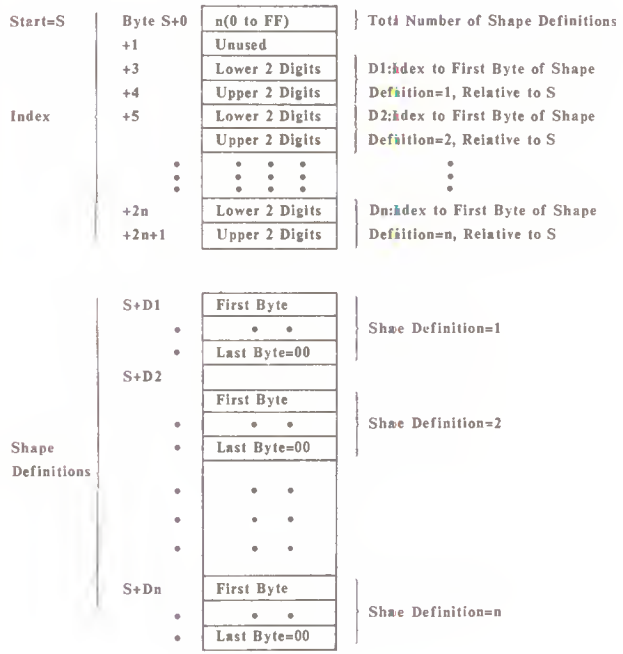


Figure 3. General Format of a Shape Table.

3.11.4 Before using DRAW, XDRAW, ROT and SCALE

Before you can use any one of the following commands: DRAW, XDRAW, ROT and SCALE, make sure you have done the followings:

1 Entered the shape table
2 Told the computer where the shape table is

Item 1 has been discussed in the previous pages. Item 2 is a very simple task; just enter the starting address of the shape table into hex location $E8 (lower two digits) and $E9 (upper two digits).

For example, if your shape table resides from address $1000 on, you can enter the computer's monitor (CALL-151) and type the following:

*E8:00 10 <RETURN>

Type CTRL-C and RETURN to go back to BASIC. Computer is now ready to interpret your shape command.

As the vectors can only point to either the left or right or up or down, diagonal lines must be approximated by a number of them which, taken together, give the impression of a diagonal line. This is shown below:

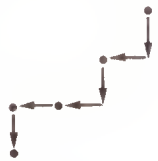


Figure 4. Vectors for a diagonal line

3.11.5 Entering a shape table

Once you have defined your shape, and broken it down into vectors, the next step is to convert the vector into binary codes so that your computer can accept them and reproduce the shape on the display later.

Two types of vector are possible: 1. move and plot; and 2. move but do not plot.

For each of these two basic types there are four directions: up, down, left, and right. In all there are eight shape vectors, and they have the following three-bit binary codes:

000	move up
001	move right
010	move down
011	move left
100	move up and plot
101	move right and plot
110	move down and plot
111	move left and plot

For instance, the diagonal line can be represented as follows, starting from the left:

100	move up and plot
101	move right and plot
100	move up and plot
101	move right and plot
101	move right and plot
100	move up and plot

The shape table in the computer's memory is made up of separate bytes, which means that only two complete vectors - of three bits each - and an incomplete vector - of only two binary bits -can be stored in each byte as there are only 8 bits within a single byte.

These incomplete vectors are movement without plotting, and they are the only ones possible in this part of the shape table byte. As this is the case, unless you can arrange your shape such that the non-plotting vectors are the very third vector in it, you should set these two bits to zero (i.e. unused).

---

e.g.] 10 DRAW 1 AT 140, 96
draws shape 1 at screen co-ordinates (140, 96)

---

e.g.
```
] 10 SCALE = 2
] 20 ROT = 32
] 30 DRAW 2 AT 40, 40
] 40 FOR D = 1 TO 2000 : NEXT D
] 50 XDRAW 2 AT 40, 40
```

This program draws shape 2 in reverse direction (i.e. rotated 180°) and in double size at (40, 40). Then wait for a while and clear the shape from the screen.

3.12 END

Purpose: Finishes program execution and returns you to command level.

Format: END

Comments: This command may be placed anywhere in your program - though it may not be all that useful as the first statement of your program.

END is the most orderly way to stop your BASIC program when it has done what you require.

This is because, unlike the similar command STOP, it does not cause a BREAK mesage to be displayed.

However, END is not essential at the end of a BASIC program - you will be returned to command level when it finishes anyway.

Example:
```
] 60 IF FIN < 0 THEN GOTO 80
] 70 END
] 80  .
     .
     .
```

In this example, if FIN is less than zero, then the program branches to line number 80.

If FIN is equal to or greater than zero, then the END command is executed, and the program terminates.

3.13 FLASH

Purpose: To cause all computer messages to alternate between character and background colour.

Format: FLASH

Comments: FLASH causes the display to alternate between NORMAL display mode and INVERSE display mode.

3.14 FOR . . . NEXT

Purpose: Loops around a group of instructions a specified number of times.

Format: **FOR** *variable* = *n* **TO** *m* [**STEP** *i*]
NEXT [*variable*]
[,*variable*] . . .

Comments: The *variable* - which is optional with the NEXT - acts as a counter for the number of times the instructions within the loop surrounded by the FOR and NEXT are executed.

n is the initial value of the counter, *m* is the final value of the counter, and *i* is the step or increment.

All the instructions in the loop are executed down to the NEXT.

Then the counter is incremented by *i*. (If you do not give a value for i, the BASIC interpreter assumes *i* is one.)

Then a check to see whether the value of the counter is greater than *m* follows. If it is not, the loop is gone through again.

If it is greater than *m*, then the program continues with the instructions that follow the NEXT statement.

The value of *i* can also be negative, in which case it is as though *m* and *n* are exchanged for their positive roles.

In other words, *n* is greater than *m*, and the counter is reduced each time through the loop until *n* is less than *m*.

FOR . . . NEXT loops can be written inside each other, or nested. In these cases, the variable names must be different, and each FOR must be matched with its corresponding NEXT.

Alternatively, one NEXT can serve a number of FORs, when it is given as NEXT *variable1, variable2, variable3* etc.

Examples:

```
] 10 FOR N = 2 TO 100 STEP 2
] 20 PRINT N/2
] 30 NEXT
```

This would print out the numbers from 1 to 50.

```
] 100 FOR N = 100 TO 2 STEP -
2
] 110 PRINT N/2
] 129 NEXT
```

This would also print out the numbers between 1 and 50, but in reverse order to the first example.

```
] 200 FOR K = 1 TO 2
] 210 FOR L = 1 TO 5
] 220 PRINT K * L; "  ",
] 230 NEXT L, K
```

This would print out the numbers:

```
1  2  3  4  5  2  4  6  8  10
```


3.15 GET

Purpose: To read a character from the keyboard without echoing it on the screen. No carriage return is necessary.

Format: GET *variable*

Comments: The *variable* may be a string or an arithmetic variable.

When the program expects an arithmetic variable and a non-numeric key is pressed, the "Syntax error" message will result.

Example:

```
] 10 GET A$
] 20 C$ = C$ + A$
] 30 PRINT C$
] 40 GOTO 10
```

3.16 GOSUB . . . RETURN

Purpose: To direct th program flow int, and return fom, a subroutine.

Format: **GOSUB** *lineumber*

•
•
•

RETURN

Comments: A subroutin may be called an number of tmes from within a program, an it is possible o call anothe subroutine fron within a suroutine, which, n turn, ma call anoth subroutine. Nesting f subroutines can be 25 leves deep. The *lienumber* needed n the GOSUB statement is th first line of he subroutine.

The RETURN statemeit terminates te execution of th subroutine, and returns th interpreter to the lin immediatelyfollowing the mot recent GOSB statement.

However, thre is no way tht the interpreer can distinguish between a subroutine and ordinary prgram lines. So, o avoid execuing the subroutie when it is not required, yu should put GOTO, STOP, r END in the ine before it starts

EExample:

```
] 10 INPUT A
] 20 GOSUB 50
] 30 PRINT A
] 40 END
] 50 IF A < 100 THEN 80
] 60 A = A + 50
] 70 RETURN
] 80 A = A + 200
] 90 RETURN
] RUN
? 40
240
] RUN
? 170
220
```

3.17 GOTO

Purpose: To direct the program flow to another part of the BASIC program.

Format: **GOTO** *linenumber*

Comments: GOTO takes your BASIC program out of its normal sequence - one line following the other - and continues execution at a point either many lines ahead, or many lines behind the line containing the GOTO.

If the line the GOTO refers to is a REMark or DATA line - which is not executable, then the instruction executed is the next executable line after *linenumber*.

GOTO can be very handy in debugging programs. You can use it in direct mode to enter a program at a certain point, rather than having the program run through from its beginning.

Example:

```
] 10 INPUT A$
] 20 B$ = B$ + A$
] 30 PRINT B$
] 40 GOTO 10
] RUN
? T
T
? H
TH
? I
THI
? S
THIS
?
```

3.18 GR

Purpose: To set up the low resolution graphics modes.

Format: **GR**

Comments: GR sets up the mixed text and low resolution graphics mode. This mode has a resolution of 40 pixels by 40 pixels and four lines of text at the bottom of the screen.

The command will clear the screen, displays the primary low resolution graphics page. The cursor will be placed just under the graphics screen, i.e., the first line from the bottom of the text screen.

3.19 HCOLOR

Purpose: To set color of subsequently plotted high resolution graphics.

Format: **HCOLOR** = *color code*

Comments: For high resolution graphics, the colors given by color code are as follows:

Code	Color
0	black
1	green
2	magenta
3	white
4	black
5	red
6	blue
7	white

Example:

```
] 10 HGR2
] 20 FOR I = 0 TO 279
] 30 HCOLOR = I/40 + 1
] 40 HPLOT I, 0 TO I, 191
] 50 NEXT I
```

Run this program and you will see 7 colored vertical bars.

3.20 HGR HGR2

Purpose: To set up the high resolution graphics modes.

Format: **HGR**
HGR2

Comments: HGR sets up the computer's mixed text and high resolution graphics mode, which has a resolution 280 pixels by 160 pixels and four lines of text at the bottom of the screen.

The command will clear the screen, displays the primary high resolution graphcis page. The cursor will be placed just under the graphics screen, i.e. the first line from the bottom of the text screen.

HGR2 has much the same effect, except that the resolution becomes 280 by 192. Text display is not available in this mode. HGR2 displays the secondary page.

3.21 HIMEM:

Purpose: To set the highest memory location available to a BASIC program.

Format: **HIMEM** : *address*

Comments: This command is used to protect the area of memory above a program for data or machine language routines.

The *address* must be in the range -65535 to 65535.

3.22 HLIN

Purpose: To draw a horizontal line in low resolution graphics.

Format: **HLIN** *x1*, *x2* AT *y*

Comments: When executing this command, a horizontal line with color defined most recently (by the COLOR command) will be drawn. The line starts at *x1* and ends at *x2* positioned at Y-coordinates *y*. *x1* and *x2* range from 0 to 39. *y* is from 0 to 47.

Example:

```
] 10 GR
] 20 COLOR = 9
] 30 HLIN 0, 39 AT 0
```

An orange line will be drawn at the top of the display screen.

3.23 HOME

Purpose: To clear screen and position the cursor at the upper left corner of the text display window.

Format: **HOME**

Comments: Characters outside the display window will be cleared. The cursor returns to the home position. Display contents beyond the text display window remain unchanged.

Example:

```
] HOME
```

All characters in the text display window will be cleared. The cursor returns to the home position. Display contents beyond the text display window remain unchanged.

3.24 HPLOT

Purpose: To draw either lines or dots in high resolution graphics.

Format:
HPLOT *x1, y1*
HPLOT TO *x1, y1*
HPLOT *x1, y1* TO *x2, y2* [, TO *x3, y3* . . .]

Comments: The first form of this command causes a dot to be plotted at the position given by *x1, y1* coordinates.

The second form causes a line to be drawn from a previously specified plotted dot to the position given by the *x1, y1* coordinates.

The third form of HPLOT draws lines from point to point as given by the pairs of (*x, y*). *x* ranges from 0 to 279 in high resolution graphic modes and 0 to 559 in double high resolution graphics mode. *y* ranges from 0 to 191 for all graphic modes.

Example:
```
] 10 HGR2
] 20 HCOLOR = 1
] 30 HPLOT 0, 0 TO 279,
191
```

This program will plot a green line from the top left-hand corner to the bottom right-hand corner of the screen.

3.25 HTAB

Purpose: To move the cursor a given number of places to the right of the left margin.

Format: **HTAB** (*displacement*)

Comments: *Displacement* ranges from 1 to 255. If *displacement* is greater than the display window width, then the cursor simply wraps around to the left-most of the same line.

Example:
```
] 10 HOME
] 20 HTAB (20)
] 30 PRINT "20 HORIZONTAL
DISPLACEMENTS"
```

3.26 IF . . . GOTO and IF . . . THEN . . .

Purpose: To direct program flow depending on the result of an evaluation.

Format: **IF** *expression* **GOTO** *linenumber*
IF *expression* **THEN** *statement*

Comments: If the *expression* is true, the *linenumber* following GOTO or the *statement* following THEN is executed, otherwise it is ignored and the program continues with the next line.

Example:

```
] NEW
] 10 INPUT A, B
] 20 IF A < B GOTO 50
] 30 PRINT A; " IS
   LARGER THAN " ;B
] 40 GOTO 10
] 50 PRINT A; " IS
   SMALLER THAN " ;B
] 60 GOTO 10
] RUN
? 32, 22
32 IS LARGER THAN 22
? 40, 90
40 IS SMALLER THAN 90
```

In statement 20, A is compared with B. If A is smaller than B, statement 50 will be executed; otherwise program continues to statement 30.


```
] NEW
] 10 INPUT A
] 20 IF A > B THEN
   B = A
] 30 PRINT B; "IS THE
   LARGEST"
] 40 GOTO 10
] RUN
? 37
37 IS THE LARGEST
? 40
40 IS THE LARGEST
```

The above program will print out the largest number so far entered.

3.27 IN#

Purpose: To accept input form a selected input device.

Format: IN# *device no.*

Comments: The number given in *device no.* must be between 0 and 7. This number determines which device your computer will expect input from.

Example:
```
] IN # 0
```

This command changes input from a peripheral device to the keyboard.

3.28 INPUT

Purpose: Allows you to enter values from the keyboard while a program is executing.

Format: INPUT ["*prompt*";] *variable 1* [, *variable 2* . .]

Comments: When the BASIC interpreter comes across an INPUT statement it displays either the "*prompt string*", or it just displays a question mark if the "*prompt string*" has not been included in the statement. Only one prompt string is allowed and it must appear immediately after INPUT.

Example:
```
] 10 INPUT "A = ";A
] 20 INPUT B
] 30 PRINT "A = ";A;"B = ";B

] RUN
A = 10
? 20

A = 10 B = 20
]
```

3.29 INVERSE

Purpose: To reverse the character and background color of all characters displayed.

Format: INVERSE

Comments: Only text displayed after the INVERSE command has been executed will be in inverse mode.

Example:

```
] 10 INVERSE
] 20 PRINT "INVERSE"
] 30 NORMAL
] 40 PRINT "NORMAL"
] RUN
```

INVERSE

NORMAL

3.30 LEFT$

Purpose: Returns a specified number of characters from the left-hand side of a character string.

Format: LEFT$ (*string$, n*)

Comments: The number *n* must be between 1 and 255, and if it is greater than the length of *string$*, then the LEFT$ function will return the entire string *string$* to the program.

LEFT$ works similarly to the RIGHT$ and MID$ string functions.

Example:

```
] 10 A$ = "Computer"
] 20 B$ = "is ok".
] 30 PRINT LEFT$(B$, 1)
] 40 PRINT LEFT$ (A$, 8)
] RUN
i
Computer
]
```

3.31 LET

Purpose: To assign a value to a variable

Format: **[LET]** *variable = expression*

Comments: LET is an optional statement, and is becoming less frequently used.

The equal sign "=" has exactly the same effect as LET.

The expression can be either a constant or an arithmetic expression.

If you attempt to assign a numeric value to a string variable, then the message "TYPE MISMATCH ERROR" will be displayed.

Example:

```
] 10 LET A = 10
] 20 PRINT A
] 30 LET B = 40
] 40 LET B = A
] 50 PRINT B
] RUN
10
10
```


3.32 LIST

Purpose: To display on the screen the BASIC program that is currently in memory.

Format: **LIST** [*linenumber 1*] [,] [*linenumber 2*]
LIST [*linenumber 1*] [-] [*linenumber 2*]

Commentts: If the *linenumber(s)* is (are) omitted, then LIST causes the entire program to be displayed.

If "*linenumber 1-*" or "*linenumber1,*" is used, then LIST will display the program from that line to the end of the program.

If "*linenumber 1, linenumber 2*" or "*linenumber 1-linenumber 2*" is used, then LIST displays only those program lines in the range given by them, inclusively.

If ", *linenumber 2*" or "-*linenumber 2*" is used, then LIST shows the lines from the beginning up to and including *linenumber 2.*

In all cases when you use *linenumber 1* or *linenumber 2*, they must be less than 63,999.

If you use just "*linenumber 1*" by itself, then just that line - if it exists - will be displayed.

3.33 LOMEM:

Purpose: To set the lowest memory location available to a BASIC program.

Format: **LOMEM** : *address*

Comments: This command is used to protect the area of memory below a program for data or machine language routines.

The *address* must be in the range -65535 to 65535.

3.34 MID$

Purpose: To return a specific number of characters from within a given string.

Format: **MID$** *(X$, i[,j])*

Comments: Both *i* and *j* must be between 1 and 255. MID$ returns *j* characters of string *X$* starting from the *i*th character.

If *j* is not specified, then MID$ has the same effect as the RIGHT$ *(X$, i)* function.

Also, if *i* is greater than LEN(*X$*), then a null string is returned.

Example:

```
] 10 X$ = "Program  in"
] 20 Y$ = "  Fortran  Basic
Cobol"
] 30 PRINT X$; MID$
(Y$, 11, 8)
] RUN
] Program in Basic
```

3.35 NEW

Purpose: Clears the current program from memory and clears all variables associated with it.

Format: NEW

Comments: This command is most commonly used to free memory before entering a new program into the computer.

BASIC returns to command level after executing NEW.

Example:
```
] NEW
]
```


3.36 NORMAL

Purpose: To return the video display from either inverse or flashing modes to the default mode.

Format: NORMAL

Comments: NORMAL sets the display with white characters on a dark background.

3.37 NOTRACE

Purpose: To stop program statement numbers from being displayed as a program executes.

Format: **NOTRACE**

Comments: This turns off TRACE. If TRACE is not on, NOTRACE has no effect.

3.38 ONERR GOTO

Purpose: To avoid halting the program when an error is encountered.

Format: **ONERR GOTO** *linenumber*

Comments: Using this statement facilitates error trapping, as it can direct the program to a routine (an error handling routine) dealing with error conditions that may arise in your program.

The RESUME statement can be used to come out from the error trapping routine.

The ONERR GOTO statement may be located anywhere within the program but it is a good practice to have it as early as possible, as this statement must be executed before the occurance of an error to avoid program interruption.

Example:

```
] 10 ONERR GOTO 100
] 20 GET A
] 30 PRINT A
] 40 GOTO 20
] 100 PRINT "INTEGERS
  ONLY"
] 110 RESUME
] RUN
1

2

INTEGERS ONLY

3
```


3.39 ON . . . GOSUB and ON . . . GOTO

Purpose: To direct the program flow depending on the value of an expression.

Format: **ON** *expression* **GOSUB** *linenumber 1* [, *linenumber 2 . . .*]
ON *expression* **GOTO** *linenumber 1* [, *linenumber 2 . . .*]

Comments: The value of the *expression* must always be an integer less than or equal to 255. When it is evaluated, it directs program flow to the corresponding line number in the list following either the GOSUB or the GOTO statements.

For instance, if the *expression* comes to five, then the program will branch to the fifth line number, and if it comes to nine, it will go to the ninth line number.

Example:

```
] 10 INPUT X
] 20 ON X GOSUB 100,
  200, 300
] 30 END
] 100 PRINT "Start of
  subroutine for X=1"
] 150 RETURN
] 200 PRINT "Start of
  subroutine for X=2"
] 250 RETURN
] 300 PRINT "Start of
  subroutine for X=3"
] 350 RETURN
] RUN
? 2

Start of subroutine for X=2
]
```


3.40 PDL

Purpose: To return the current value of the game adapter.

Format: **PDL** (*n*)

Comments: The value of *n* specifies which game paddle to be read and may be 0 or 1.

The value returned ranges from 0 to 255 (decimal).

Example:

```
] 10 PRINT PDL (0), PDL (1)
] 20 GOTO 10
] RUN
165  206
194  35
```

3.41 PEEK

Purpose: To read the byte at a specified memory location.

Format: PEEK (*i*)

Comments: *i* must be an integer in the range 0 to 65535. The byte returned by PEEK will be an integer between 0 and 255.

Example:
```
] I=PEEK (48345)
```


3.42 PLOT

Purpose: To draw dots in low resolution graphics.

Format: PLOT *X*, *Y*

Comments: This command causes a dot to be plotted at the position given by x, y coordinates in the low-resolution graphics. The value of *X* is from 0 to 39 and the value of *Y* is from 0 to 47.

Example:
```
] 10 GR
] 20 COLOR =3
] 30 PLOT 0, 0
```
A purple dot will be plotted on the top left corner of the screen.

3.43 POKE

Purpose: To write a byte of data into a specified memory location.

Format: **POKE** *n*, *m*

Comments: The data to be placed in memory is *m*, which must be between 0 and 255. The memory locations is *n*, and this must be in the range 0 to 65,535.

Important: The computer does not check on the address you use in the POKE command, so if you POKE a value into one of its dedicated memory areas, or into your BASIC program area, you may find that the machine ceases to operate.

Example:

```
] POKE 1000, 10
```


3.44 POP

Purpose: To change the action of a RETURN from a subroutine.

Format: **POP**

Comments: POP effectively removes the top address from the stack of subroutine's RETURN addresses.

Example:

```
] 10 GOSUB 100
] 20 END
] 100 GOSUB 200
] 110 PRINT "THIS
  STATEMENT NEGLECTED"
] 120 RETURN
] 200 POP
] 210 RETURN
]
RUN
```

Program flows from statement 10, 100, 200, 210, 20. Statement 110 is skipped due to pop action.

3.45 PR#

Purpose: To switch the output to the selected device.

Format: **PR#** *device no.*

Comments: The number given as *device no.* must be between 0 and 7. If there is nothing connected at the given device then your computer will suspend operation, and you will have to RESET the machine.

PR#0: turn off all selected device. Set output device to display monitor.

PR#1: characters are output to the parallel or serial printer (selected by the switch on the front panel).

PR#2: characters are output to the serial interface 2.

PR#3: switch to 80 column display.

PR#4: activate the mouse interface.

PR#5: activate the built-in expansion RAM interface or the interface card plugged into the optional expansion box.

PR#6: characters are output to the disk controller port, which in turn activate the built-in disk drive.

PR#7: activate the built-in 3.5" disk drive interface or the interface card plugged into the expansion connector.

3.46 PRINT

Purpose: To display characters on the display screen.

Format: **PRINT** [*list*] [;] [,]
? [*list*] [;] [,]

Comments: The *list* is a number of values - either variables or constants - which may be strings or numbers.

If literal strings are to be printed out, they must be enclosed by quotation marks ("literal").

If the list is not given, then PRINT will output a blank line, which can be handy for spacing out results as you display them on the screen.

If you separate the values in the list by commas, then each value will start in the next tab field, each of which comprises 16 columns.

If you separate the values by semi-colons, then the values will be displayed continuously.

PRINT will use either integer or fixed point format for outputting numbers depending on whether they are expressible in nine or fewer digits.

Example:

```
] PRINT 10, 20
10          20
] PRINT 10; 20
1020
] PRINT "HI MOM"
HI MOM
]? "YOU LOOK TERRIFIC"
YOU LOOK TERRIFIC
```

3.47 READ

Purpose: To read values from a DATA statement and to assign them to variables.

Format: **READ** *variable* [, *variable* . . .]

Comments: The READ statement must be accompanied by the DATA statement. Enough data must be specified by the DATA statement in order to be READ, otherwise an 'OUT OF DATA ERROR' may result.

variable can be either numeric or string variables.

DATA statements can be re-used after they have been READ once, but to do this you must use the RESTORE command.

Example:

```
] 10 READ A
] 20 READ B$
] 30 PRINT A; "   "; B$
] 40 DATA 10, IS TEN

] RUN
10 IS TEN
```


3.48 REM

Purpose: To let you REMind yourself by REMarks of what your program intends to do.

Format: **REM** *remark*

Comments: REM statements are not executed, and they only appear when your BASIC program is listed. You will find them useful to document your programs with REMs, despite the fact that they take up memory space.

They can be added at the end of BASIC program lines if they are preceded by a colon.

Example:

```
] 10 REM THIS IS A REMARK
] 20 PI = 3.14 : REM
APPROXIMATE VALUE OF PI
```

3.49 RESTORE

Purpose: To use DATA values again after they have been READ.

Format: **RESTORE**

Comments: After a RESTORE statement is executed, the next READ statement will read the first item of the very first DATA statement in your program.

Example:

```
] 10 READ A
] 20 READ B
] 30 DATA 10, 20, 30
] 40 PRINT A, B
] 50 RESTORE
] 60 READ C, D, E
] 70 PRINT C, D, E

] RUN
10  20
10  20  30
```


3.50 RESUME

Purpose: To restart a program that has been halted due to an error.

Format: **RESUME**

Comments: This command is mainly used at end of an error handling routine, and it causes the program to restart execution at the statement which caused the error. If an error occurs during the error handling, then RESUME will place your program in an infinite loop. You can get out of this only by pressing the RESET botton.

3.51 RIGHT$

Purpose: To return a specified number of characters from a string proceeding from the right.

Format: RIGHT$(*X$, i*)

Comments: If *i* is greater than or equal to LEN(X$), then the whole string is returned. Integer *i* must be between 1 and 255. See LEN, LEFT$ and MID$ string functions.

Example:

```
] 10 X$ = "COMPUTER IS OK"
] 20 PRINT RIGHT$ (X$, 2)
] RUN
OK
]
```


3.52 ROT

Purpose: To specify the angle by which a shape drawn by either DRAW or XDRAW commands will ROTate.

Format: **ROT** = *angle*

Comments: For shapes drawn using the DRAW shape commands, the value given as angle can be anything between 0 and 255, with 255 representing a 360° rotation.

For shapes drawn using the shape table method, a value of 16 for angle will rotate a shape through one right angle (90°); 32 will rotate it through two right angles (180°); 48 will rotate it through three right angles (270°); and 64 (=4 x 16) will perform a complete rotation, bringing it back to its original position.

Example: See DRAW command for an example on ROT.

3.53 RUN

Purpose: To start a program execution.

Format: RUN [*linenumber*]

Comments: Unless *linenumber* is given, RUN always begins execution from the very beginning of a program.

When *linenumber* is specified, it starts at that line.

Example:

```
] 10 PRINT "FIRST LINE"
] 20 PRINT "SECOND LINE"
] 30 PRINT "ALL DONE"

] RUN
FIRST LINE
SECOND LINE
ALL DONE
] RUN 30
ALL DONE
```


3.54 SCALE

Purpose: To increase or decrease the size of shapes created by DRAW or XDRAW.

Format: SCALE = *size*

Comments: The *size* number must be between 0 and 255. The default value 0 yields the highest magnification while 1 gives you the smallest possible shape.

3.55 SCRN

Purpose: To find out the color code of a point in low resolution graphics.

Format: SCRN (*x*, *y*)

Comments: This command returns the color code of point (*x*, *y*) on the low resolution graphics screen.

x is from 0 to 39 while *y* is from 0 to 47. The color code ranges from 0 to 15.

Example:

```
] 10 GR
] 20 COLOR = 15
] 30 PLOT 0, 40
] 40 PRINT SCRN (0, 40)
] RUN
15
```


3.56 SPC

Purpose: To separate two printed items by a specified number of spaces.

Format: **PRINT SPC** (*expression*)

Comments: This command, which is used in conjunction with the PRINT command, can be used to layout results printed by a program.

The value evaluated from expression must range between 0 and 255.

Example:

```
10 PRINT "A"; SPC(20); "B"
] RUN
A                    B
```

3.57 SPEED

Purpose: To specifyy the rate which characters aare to be sent to an output devicce.

Format: **SPEED** = *ratte*

Comments: The slowestt rate is zero. The fastest and| the default rate is 255.

Example:

```
] 10 SPEED) = 10
] 20 PRINTT "THIS IS SLOW
  SPEED"
] 30 SPEED) = 255.
] 40 PRINTT "THIS IS HIGH
  SPEED"
```


3.58 STOP

Purpose: To halt program execution and return to command level.

Format: **STOP**

Comments: This command is similar to END, except that STOP causes the message "BREAK IN nnnnn" to be displayed, where nnnnn is the line number of the STOP statement.

The BASIC interpreter always returns to the command level after a STOP is executed.

Example:

```
] 10 READ A
] 20 PRINT 7 * A
] 30 STOP
] 40 DATA 7

] RUN
49
BREAK IN 30
```

3.59 STR$

Purpose: To return a string representation of a numeric value.

Format: STR$(*x*)

Comments: This is a good means of checking the number of digits in a numeric constant, if it is used in conjunction with the LEN string function.

Example:

```
] 10 INPUT A
] 20 X$ = STR$ (A)
] 30 PRINT X$
] 40 PRINT "THE NUMBER
  HAS  "; LEN (X$); "  DIGITS"
] 50 GOTO 10
] RUN
? 1234
1234
THE NUMBER HAS 4 DIGITS
?
```


3.60 TAB

Purpose: To specify a number of places to the right of the left margin for the cursor.

Format: PRINT TAB (*expression*)

Comments: The TAB function only moves the cursor to the right. Hence, if the value evaluated from the expression is smaller than the column number of the current cursor position, the cursor will not move. The value of *expression* must be from 0 to 255.

Example:

```
] 10 PRINT TAB (10)
] 20 PRINT "10 COLUMNS TO
THE LEFT"
```

The computer will print out the string on statement 20 starting from the 10th column.

3.61 TEXT

Purpose: To set the display to full-screen text mode.

Format: TEXT

Comments: In the full-screen text mode, only characters (no graphics) are displayed in 24 rows by 40/80 columns.

TEXT set the display to full-screen text mode.

3.62 TRACE

Purpose: To display line numbers of a program as each line is executed.

Format: TRACE

Comments: TRACE is very useful in determining where a program may be going wrong (debugging). The line number of statements executed thereafter is displayed. TRACE is turned off by the NOTRACE command.

Example:

```
] 10 FOR J = 1 TO 3
] 20 PRINT J * 2
] 30 NEXT J
] 40 END
] TRACE
] RUN

# 10        # 20  2
# 30        # 20  4
# 30        # 20  6
# 30  # 40
]
```

3.63 USR

Purpose: This command specifies a parameter of an assembly language subroutine.

Format: USR (*n*)

Comments: *n* is an arithmetic expression. When USR is encountered, the arithmetic expression is evaluated and placed in the floating point accumulator, and a JSR to location $0A is performed which must then contain a JUMP to the beginning location of the machine-language subroutine. An RTS machine instruction should be executed at the end of the machine language subroutine.

Example:

```
] CALL-151
* 0A : 4C  10  03
* 310 : 6C
* E003G
] PRINT USR (9) * 12
108
]
```

A JUMP $310 instruction is placed at location $0A and RTS instruction at $310.

3.64 VLIN

Purpose: To draw a vertical line in low resolution graphics.

Format: VLIN *y1*, *y2*, AT *x*

Comments: In low resolution graphics, a vertical line will be drawn from *y1* to *y2* positioned at X-coordinates *x*. *y1* and *y2* ranges from 0 to 47 and *x* is from 0 to 39.

Example: Refer to example of COLOR.

3.65 VTAB

Purpose: To move the cursor a given number of lines down the display screen.

Format: VTAB *number*

Comments: As there are only 24 lines on the display, *number* values outside 1 to 24 will cause an error. The screen lines are numbered from top to bottom.

Exmaple:
```
] 10 HOME
] 20 VTAB 10
] 30 PRINT "DOWN 10 ROWS"
```


3.66 WAIT

Purpose: To suspend a program's execution while watching the status of an input port.

Format: WAIT *portnumber, n* [*,m*]

Comments: The command suspends a program's execution until a specified input port develops an expected bit pattern.

The command loops around, reads the data at the port, XORs it with the integer value *m*, and then ANDs the result with the integer value *n*. If *m* is not specified, it is taken to be zero.

If the result at the end of the loop is zero, the loop starts over again.

If the result is not zero, then the program resumes execution at the next executable statement after WAIT.

Careful: You can get into a continuous loop with the use of the WAIT command. Warm start the computer by pressing CTRL + RESET if you believe this has happened. It will let you out of the loop, and return you to command level.

Exmaple: WAIT 49152, 128
This will wait until a key is pressed which will set the most significant bit.

3.6' XDRAW

Purpose: To draw or to erase a defined shape.

Format: **XDRAW** *shape no.* **At** *x, y*

Comments: This command allows you to draw a shape if it is not already on screen, and erase it if it is.

Example:
```
] 10 DRAW 1 AT 100, 100
] 20 FOR D = 1 TO 1000
  : NEXT D : REM DELAY
] 30 XDRAW 1 AT 100, 100
```

Assuming you have defined shape 1, this program will first draw it at co-ordinates (100, 100), then wait for a while and finally erase the drawn shape.

CHAPTER 4

BASIC FUNCTIONS

4. BASIC FUNCTIONS

This chapter lists alphabetically and descoribes the intrinsic functions available for the computer's BASIC.

The arguments - or parameters - for the fumctions are usually enclosed in parentheses.

The conventions followed for the argumemts are as follows:-

x and *y*	Represent any numeric expressioons
i and *j*	Represent any integer expressioms
X$ and *Y*$	Represent any string expressionss

4.1 ABS

Purpose: To give the absolute value of a numeric expression.

Format: ABS (*x*)

Comments: This function always returns a positive value, and can be used with either floating point or integer values.

Example:
```
] PRINT ABS(9 * (-7))
63
]
```


4.2 ASC

Purpose: To return the ASCII code for the first character of aa specified string.

Format: ASC (*X$*)

Comments: An error will result if ' the string specified is a null strimng.

Example:
```
] PRINT ASC ("LASERR")
76
]
```

4.3 ATN

Purpose: To calculate the arctangent of a value.

Format: ATN (x)

Comments: This gives the arctangent of x in radians, with the result in the range $-\pi/2$ to $\pi/2$.

Example:
```
] PRINT ATN(8)
1.44644133
]
```


4.4 COS

Purpose: To calculate the cosine of an angle.

Format: COS (x)

Comments: The value of the angle is in radians, and not degrees.

Example:
```
] PRINT COS(2)
-.416146836
]
```

4.5 EXP

Purpose: To calculate the value of "e" - the base of natural logarithms - raised to a specified power.

Format: **EXP** (*x*)

Comments: The value of *x* should be less than 89, or an overflow error will result.

Example:
```
] PRINT EXP(9)
8103.08393
]
```


4.6 FRE

Purpose: Repports on the number of bytes in memory that are not being useed by BASIC.

Format: **FRE** (*expression*)

Comments: Beccause strings in BASIC can havve different lengths, and need to be manipulated. This freqquently causes the memory to beccome very fragmented. Using thiss statement with a dummy argguument can force BASIC to gathher up all the loose fraggments into contiguous whooles (garbage collection).

Thiis frees up areas of memory, andd can often give you a surpprising amount more.

Example:
```
] X = FRE (0)
```
Thiis would lead to a garbage colldection operation. It may take somne time.

```
] PRRINT FRE (0)
```
In addition to a garbage colldection operation, this would prinnt out the amount in bytes of freee user memory.

4.7 INT

Purpose: To round a fractional number down to a whole number.

Format: INT (*x*)

Comments: This function always returns an integer that is less than or equal to the number *x*.

Example:

```
] PRINT INT (31.98)
31
] PRINT INT (-31.98)
-32
]
```


4.8 LEN

Purpose: To return the number of characters in a string.

Format: LEN (*X$*)

Comments: This function counts all characters in the specified string, including blanks and non-printing characters.

Example:

```
] NEW
] 30 X$ = "COMPUTER"
] 40 PRINT LEN(X$)
] RUN
8
]
```

4.9 LOG

Purpose: To calculate the natural logarithm of a specified value.

Format: **LOG** (*x*)

Comments: The value *x* must be greater than zero.

Example:
```
] PRINT LOG(669)
6.50578406
]
```


4.10 POS

Purpose:: To return the current horizontal cursor position.

Format: **POS** (*i*)

Commemts: The leftmost cursor position is 0 on the display screen. The argument *i* is a dummy.

Example:
```
] HTAB (10) : PRINT POS (1)
              9
]
```

4.11 RND

Purpose: To return a random number between 0 and 1.

Format: RND (x)

Comments: The value of the dummy argument, x, determines how the random numbers are generated. If x is greater than zero then RND (x) generates a new random number everytime it is used.

If x is less than zero, then RND (x) generates the same random number everytime it is used with the same argument.

Example:

```
] 10 FOR I = 1 TO 8
] 30 PRINT INT(RND(1) * 1000)
] 50 NEXT
] RUN

 797
 584
 268
 397
 31
 932
]
```


4.12 SGN

Purpose: To return the sign of a number.

Format: SGN (x)

Comments: If the number is greater than zero, then SGN returns 1; if it is zero, SGN returns zero; and if it is negative, then SGN returns -1.

Example:

```
] 10 INPUT A
] 20 B = SGN (A)
] 30 IF B = 0  THEN 90
] 40 IF B > 0  THEN 70
] 50 PRINT "A IS NEGATIVE"
] 60 GOTO 10
] 70 PRINT "A IS POSITIVE"
] 80 GOTO 10
] 90 PRINT "A IS ZERO"
] 100 GOTO 10
] RUN
? 1
A IS POSITIVE
? -4
A IS NEGATIVE
? 0
A IS ZERO
?
```

4.13 SIN

Purpose: To calculate the sine of a specified angle.

Format: SIN (x)

Comments: The value of the angle must be given in radians and not degrees.

Example:

```
] PRINT SIN(4)
-.756802495
]
```


4.14 SQR

Purpose: To calculate the square root of a specified value.

Format: SQR (x)

Comments: A negative value for x will cause an error.

Example:

```
] 10 FOR I = 1 TO 6
] 20 PRINT 2 ^ (2 * I),
  SQR (2 ^ (2 * I))
] 30 NEXT
] RUN
4         2
16        4
64        8
256       16
1024      32
4096      64
]
```

4.15 TAN

Purpose: To calculate the tangent of a specified angle.

Format: TAN (*x*)

Comments: The value of the angle must be given in radians and not degrees.

Example:
```
] PRINT TAN (12)
-.635859926
]
```


4.16 VAL

Purpose: To return the numerical value of a specified string.

Format: VAL (*X$*)

Comments: The function ignores leading spaces of the specified string.

Example:
```
] PRINT VAL ("   78")
78
]
```

APPENDIX

APPENDIX A

SPEEDING UP PROGRAM EXECUTION IN THE LASER 128EX

A. SPEEDING UP PROGRAM EXECUTION IN THE LASER 128EX

Running at its normal speed, the LASER 128 can handle most tasks without any difficulty. However, for applications requiring a large amount of complicated computations and fast responses a demanding user may not be satisfied with the processing speed of the LASER 128.

To fulfil your needs, the LASER 128EX, equipped with a high-speed central processor, is capable of running programs at a higher rate. The execution speed is software-selectable so that it can be chosen for each program section at will.

When the computer is turned on, the default speed is the standard 1MHz. You can change the computing speed by the following methods: Holding down the numeric key "2" or "3" while turning the computer on or while pressing CTRL-RESET. The number "2" option will cause the computer to run a program at 2.3 times the normal speed. The number "3" option will make a program run up to 3.6 times its normal speed.

If either of the two "fast" modes is entered while the computer is in 40-column text mode, you can observe that the checker-board cursor blinks at a higher rate.

Moreover, if you "beep" the speaker by pressing "CTRL-G", you will notice that the pitch of the sound is higher than usual. This provides a simple means of telling which mode the computer is currently in.

APPENDIX B

INSTALLATION OF EXPANSION RAM

B. INSTALLATION OF EXPANSION RAM

Besides the 128 K-byte system RAM that comes with your computer, it also has room for accommodating up to 1 M-byte expansion RAM. To install additional RAM in the computer, you need the following items:

- Optional memory expansion card (included in the LASER 128EX). To install it in the LASER 128, please refer to the installation manual which comes with the RAM card.

- 256K x 1 bit dynamic RAM chips type (41256). The row address access time for the RAM chips should be 120 ns for the LASER 128EX and 150 ns for the LASER 128. The quantity required depends on the size of the expansion RAM and is shown as follows:

Expansion RAM size	Quantity of 41256
256 K	8 pcs.
512 K	16 pcs.
786 K	24 pcs.
1024 K	32 pcs.

If at all possible, you should consult your dealer for installation of expansion RAM. However, if you have to do it yourself, do it with care! INCORRECT INSTALLATION MAY CAUSE PERMANENT DAMAGE TO YOUR RAM CHIPS AND/OR THE COMPUTER!

To install the expansion RAM, here are the procedures to follow:

- Turn off power.

- Disconnect the computer from the AC power adaptor and any other peripherals.

- Turn the computer over and remove the top cabinet by loosening the screws at the bottom cabinet as in Figure B-1.

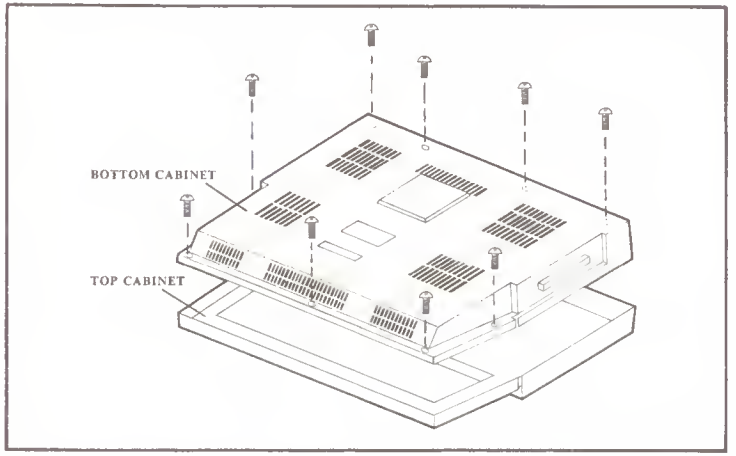


Figure B-1 Removing the top cabinet

- Turn the computer right-side-up and remove the keyboard as in Figure B-2.

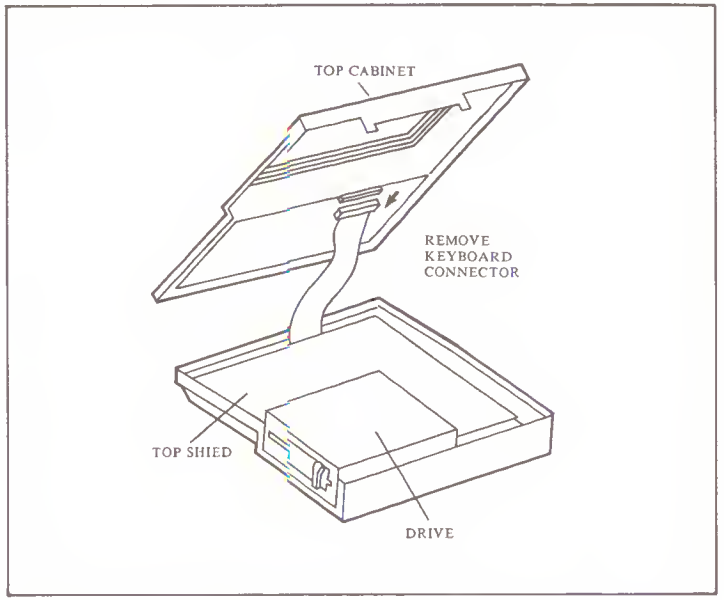


Figure B-2 Removing the keyboard

- Remove the RAM door on the top metal shield plate using a screwdriver as shown in Figure B-3.

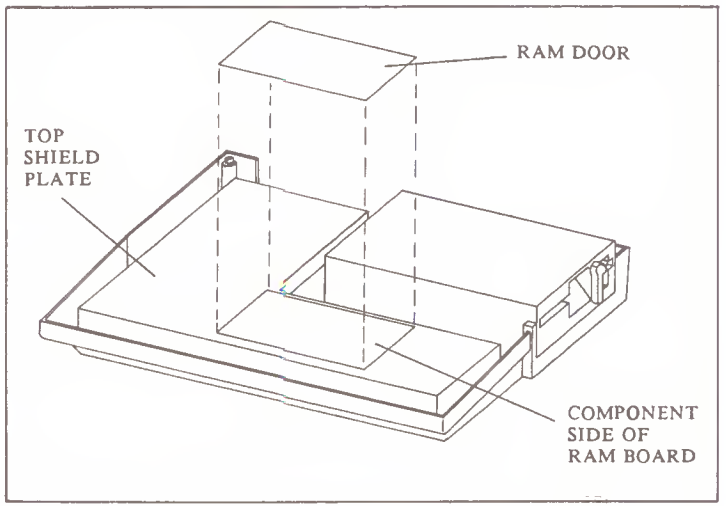


Figure B-3 Removing the top shield plate

The component side off the RAM card will then be exposed.

- Insert the dynamic RAM chips into the IC sockets on the memory expansion card carefully, paying particular attention to the orientation of the RAM chips. THE RAM CHIPS WILL BE DAMAGED IF INSERTED IN THE WRONG DIRECTION (see Figure B-4).

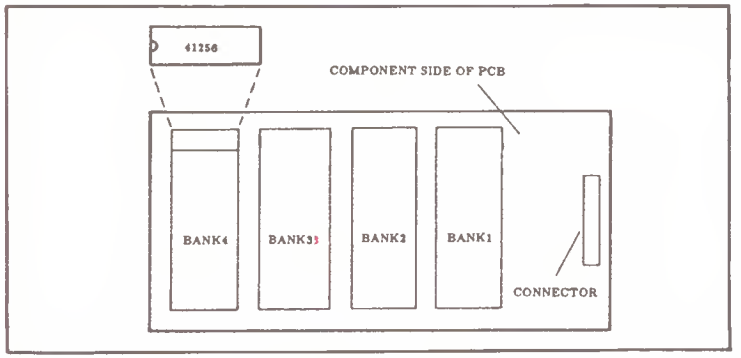


Figure B-4 Inserting the dynamic RAM chips

- The IC sockets on the memory expansion card are arranged in four rows. Each row contains eight IC socket. Expansion RAM chips must be added in groups of eight, with each group occupying one of the four rows of IC sockets. The RAM chips must be inserted in bank sequence according to the bank number indicated in Figure B-4. Inserting RAM chips in one row gives you an additional 256 K-byte expansion RAM so that up to 1 M-byte expansion RAM can be installed.

- Cover the RAM board with the RAM door and fix it on the top shield by tightening the screws.

- Put the keyboard back to its original position.

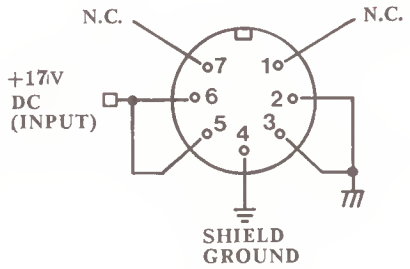
- Install the top cabinet and tighten all the screws on the bottom cabinet.

  Finally, connect the AC power adaptor and other peripheral devices back to the computer and turn it on. The computer will function as usual except that it has an extra memory area for data storage.

APPENDIX C

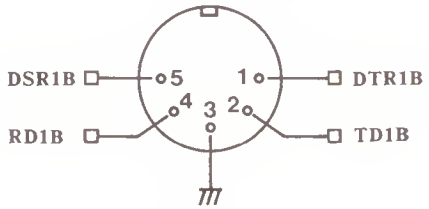
EXPANSION CONNECTORS DIAGRAMS

C. EXPANSION CONNECTORS DIAGRAMS



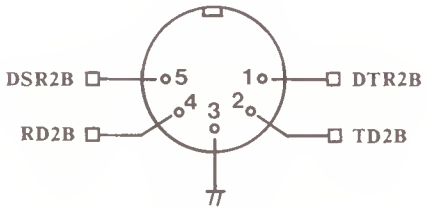
1. N.C. - no connection
2. ground
3. ground
4. shield ground
5. +17 volt input
6. +17 volt input
7. N.C. - no connection

Figure C-1 Power connector



1. DTR1B - Data Terminal ready
2. TD1B - Transmit data
3. ground
4. RD1B - receive data
5. DSR1B - data set ready

Figure C-2 Serial printer connector



1. DTR2B - data terminal ready
2. TD2B - transmit data
3. ground
4. RD2B - receive data
5. DSR2B - data set ready

Figure C-3 Serial interface connector

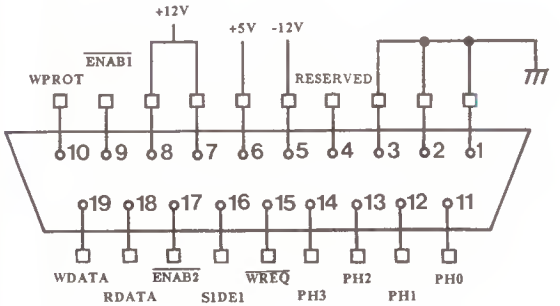


Figure C-4 External drive connector

1. ground
2. ground
3. ground
4. reserved
5. -12 volt
6. +5 volt
7. + 12 volt
8. + 12 volt
9. ENAB1 - active low when first drive is accessed
10. WPROT - write protect
11. PH0 - drive control
12. PH1 - drive control
13. PH2 - drive control
14. PH3 - drive control
15. WREQ - active low when write to drive is required
16. SIDE1 - active high when side 1 of diskette is accessed
17. ENAB2 - active low when second drive is accessed
18. RDATA - drive data input to computer
19. WDATA - drive data output from computer

Figure C-5 Video connector

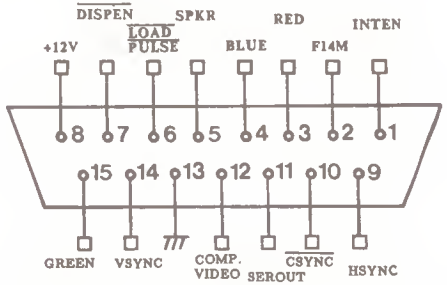


Figure C-6 Video expansion connector

1. INTEN' - intensity signal
2. F14M - 14 MHZ
3. RED-signal
4. BLUE - signal
5. SPKR - sound output
6. $\overline{\text{LOAD PULSE}}$-for LCD
7. $\overline{\text{DISPEN}}$-for LCD
8. + 12 volt
9. HSYNC - horizontal synchronization signal
10. $\overline{\text{CSYNC}}$ - composite synchronization signal (active low)
11. SEROUT -data for LCD
12. COMP VIDEO- composite video output
13. ground
14. VSYNC - vertical synchronization signal
15. GREEN-signal

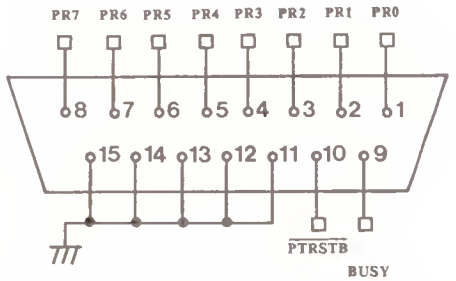


Figure C-7 Parallel printer connector

1. printer data bus
2. printer data bus
3. printer data bus
4. printer data bus
5. printer data bus
6. printer data bus
7. printer data bus
8. printer data bus
9. BUSY -from printer
10. $\overline{\text{PTRSTB}}$ - printer data strobe (active low)
11. ground
12. ground
13. ground
14. ground
15. ground

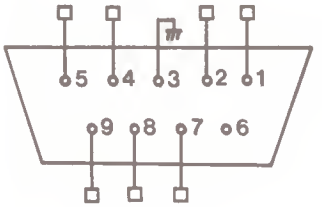


Figure C-8 Game input connector

When the port is used with a mouse:

1. MOUSE SIGNATURE
2. + 5 volt
3. ground
4. X DIR - X direction of mouse
5. X INT - X interrupt of mouse
6. N.C. - no connection
7. MOUSE BUTTON
8. Y DIR - Y direction of mouse
9. Y INT - Y interrupt of mouse

When the port is used with a paddle / joystick:

1. SW1 - game switch 1
2. + 5 volt
3. ground
4. reserved
5. GAMEO - game paddle 0
6. N.C. - no connection
7. SWO - game switch 0
8. GAME 1 - game paddle 1
9. reserved

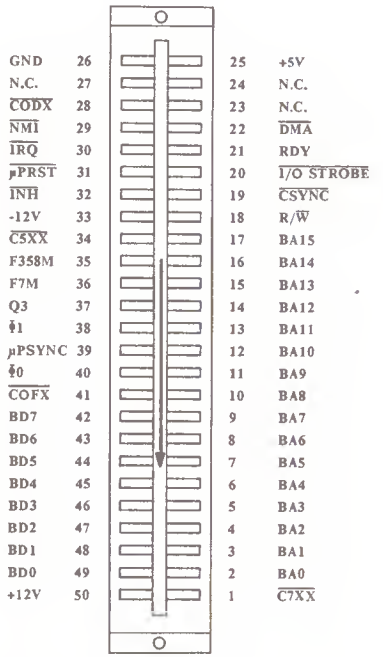


Figure C-9 Expansion connector

1. C7XX - active low when address $C7XX is accessed
2. address bus
3. address bus
4. address bus
5. address bus
6. address bus
7. address bus
8. address bus
9. address bus
10. address bus
11. address bus
12. address bus
13. address bus
14. address bus
15. address bus
16. address bus
17. address bus
18. R/W - read/write
19. CSYNC - composite sync
20. I/O STROBE - active low when slot $C800 - $CFFF is accessed
21. RDY - CPU Ready input
22. DMA - active low for direct memory access
23. N.C. - no connection
24. N.C. - no connection
25. + 5 volt
26. ground
27. N.C. - no connection
28. C0DX - active low when $C0DX is accessed
29. NMI - active low for non- maskable interrupt
30. IQR - active low for interrupt request
31. UPRST - active low during reset
32. INH - active low to inhibit memory .
33. -12 volt

34. C5XX - active low when $C5XX is accessed
35. F358M - chroma frequency signal
36. F7M - 7 MHZ signal
37. Q3 - 2 MHZ signal
38. Ø1 - 1 MHZ signal
39. UPSYNC - CPU signal
40. Øo - 1 MHZ signal
41. C0FX - active low when $C0FX is accessed
42. data bus
43. data buus
44. data buus
45. data buus
46. data buus
47. data buus
48. data buus
49. data buus
50. +12 voblt

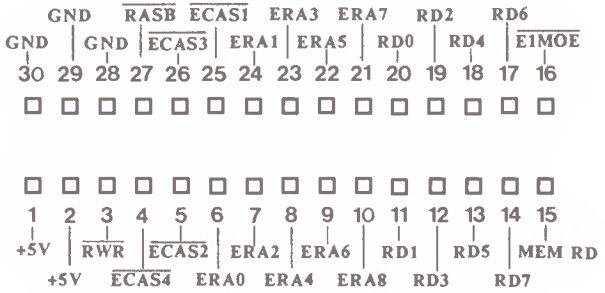


Figure C-10 Expansion memory connector

1. +5v
2. +5v
3. RWR - active low for RAM write
4. ECAS4 - Bank 4 RAM CAS
5. ECAS2 - Bank 2 RAM CAS
6. RAM address
7. RAM address
8. RAM address
9. RAM address
10. RAM address
11. RAM data
12. RAM data
13. RAM data
14. RAM data
15. MEMRD- active high for read
16. EIMOE- expansion RAM data buffer output enable
17. RAM data
18. RAM data
19. RAM data
20. RAM data
21. RAM address
22. RAM address
23. RAM address
24. RAM address
25. ECAS1 - Bank 1 RAMM CAS
26. ECAS3 - Bank 3 RAMM CAS
27. RASB - RAM RAS
28. ground
29. ground
30. ground

APPENDIX D

ERROR MESSAGES

D. ERROR MESSAGE

BASIC ERROR MESSAGES

In most cases, when an error occurs in a BASIC program, the Interpreter returns to the command level.

This is designated by the "]" prompt and a flashing cursor. The program remains in memory, and the variables are set at the values they had assumed at the time the error was encountered.

You can use the PRINT command in direct mode to ascertain the values your program variables had at the time of the error.

To avoid your program stopping on coming across an error, you can use error trapping. See the ONERR GOTO statement for an explanation of how to use this technique.

LIST OF ERROR MESSAGES AND EXPLANATIONS

CAN'T CONTINUE

This message will occur when you have halted a program (using STOP or CTRL-C or BREAK), then edit it, and try to CONTinue. It will also arise when you try to CONTinue a program after an error has occured.

DIVISION BY ZERO

This error will stop your program executing. You cannot divide a number by zero.

ILLEGAL DIRECT

This occurs when you try to use the following statements in the direct mode:

GET
DEF FN
INPUT

ILLEGAL QUANTITY

The argument given to an arithmetic or string expression either does not match the type of expression or it is out of the expression's range. Possibilities are:

Using the SQR - square root - function with a negative argument.

Using a negative argument for the subscript of an array.

Using a negative or zero argument with the LOG - natural logarithm - function.

NEXT WITHOUT FOR

Self explanatory. The variable given in a NEXT statement does not match the variable name given in a FOR statement which is in operation.

Alternatively, a NEXT does not correspond to any FOR statement which is in effect.

OUT OF DATA

This occurs when a READ statement is executed but either all the DATA statements have already been read, or there is not a match between the number of variables in the READ statement and the number of values given in the DATA statement.

OUT OF MEMORY

This message can arise from a number of conditions and errors:

- Your program is too large for the available memory.
- Your program has too many variables for the BASIC interpreter to handle - a number in excess of 100 combined with a long program may cause this error.
- If you have FOR . . . NEXT loops nested to more than 10 levels.
- If you have GOSUB . . . RETURNs nested more than 24 levels.
- If an expression is too complicated for the interpreter to decipher.
- If parentheses are nested to more than 36 levels.

The last two possibilities are related, with the second giving an indication of the level of complexity permitted.

OVERFLOW

The result of a calculation exceeds 10E38, which is the computer's maximum number size. If a number is calculated as less than 10E-28 - the computer's minimum number size - then the result becomes zero, and execution continues with no message being printed.

REDIM'D ARRAY

If an array has been used relying on the default DIMensioning of any array, and then the array is explicitly DIMensioned with another statement, this message will be displayed. Alternatively, it occurs when two different DIMension statements exist for the same array.

RETURN WITHOUT GOSUB

Self explanatory. A RETURN statement exists without a corresponding GOSUB statement.

STRING TOO LONG

Trying to use the string concatenation operator "+" to bring together two strings whose added length is greater than 255 characters. 255 is the maximum length of a string in the computer's BASIC.

BAD SUBSCRIPT

Your program has tried to refer to an array element which is greater than the size of the subscript given for the array in its DIM statement.

This can also occur if an array is referred to using the wrong number of dimensions.

For example, if array ARRAY has been DIMensioned DIM ARRAY (10, 10, 10), and a subsequent statement like ARRAY (9, 8, 7, 6) =54 is come across, then this error message will be displayed.

SYNTAX

The manner in which a statement, function, or expression has been put is incorrect. Things to look for are missing commas, spaces, parentheses, periods, or illegal characters starting a variable name.

TYPE MISMATCH

This occurs when you try to assign a string value to a numeric variable, or a numeric value to a string variable, or if either a numeric function receives a string value, or a string function receives a numeric value.

UNDEF'D FUNCTION

A reference is made to a user defined statement which does not exist in the BASIC program.

UNDEF'D STATEMENT

A line referred to in a GOTO, GOSUB, or IF . . . GOTO statement does not exist in your program.

APPENDIX E

KEYS AND THE ASSOCIATED CODES

E. KEYS AND THE ASSOCIATED CODES

KEY	NORMAL	CHAR	CONTROL	CHAR	SHIFT	CHAR	BOTH	CHAR
DELETE	7F	DEF	7F	DEL	7F	DEL	7F	DEL
←	ØB	BS	ØB	BS	ØB	BS	ØB	BS
TAB	Ø9	HT	Ø9	HT	Ø9	HT	Ø9	HT
↓	ØA	LF	ØA	LF	ØA	LF	ØA	LF
↑	ØB	VT	ØB	VT	ØB	VT	ØB	VT
RETURN	ØD	CR	ØD	CR	ØD	CR	ØD	CR
→	15	NAK	15	NAK	15	NAK	15	NAK
ESC	1B	ESC	1B	ESC	1B	ESC	1B	ESC
SPACE	2Ø	SP	2Ø	SP	2Ø	SP	2Ø	SP
'"	27	'	27	'	22	"	22	"
,<	2C	,	2C	,	3C	<	3C	<
-_	2D	-	1F	US	5F	_	1F	US
.>	2E	.	2E	.	3E	>	3E	>
/?	2F	/	2F	/	3F	?	3F	?
Ø)	3Ø	Ø	3Ø	Ø	29	)	29	)
1!	31	1	31	1	21	!	21	!
2@	32	2	ØØ	NUL	4Ø	@	ØØ	NUL
3=	33	3	33	3	23	=	23	=
4$	34	4	34	4	24	$	24	$
5%	35	5	35	5	25	%	25	%
6^	36	6	1E	RS	5E	^	1E	RS
7&	37	7	37	7	26	&	26	&
B*	3B	B	3B	B	2A	*	2A	*
9(	39	9	39	9	2B	(	2B	(
;:	3B	;	3B	;	3A	:	3A	:
=+	3D	=	3D	=	2B	+	2B	+
[{	5B	[	1B	ESC	7B	{	1B	ESC
|	5C	\	1C	FS	7C	|	1C	FS
]}	5D	]	1D	GS	7D	}	1D	GS
`~	6Ø	`	60	`	7E	~	7E	~
A	61	a	Ø1	SOH	41	A	Ø1	SOH
B	62	b	Ø2	STX	42	B	Ø2	STX
C	63	c	Ø3	ETX	43	C	Ø3	ETX
D	64	d	Ø4	EOT	44	D	Ø4	EOT
E	65	e	Ø5	ENO	45	E	Ø5	ENO
F	66	f	Ø6	ACK	46	F	Ø6	ACK
G	67	g	Ø7	BEL	47	G	Ø7	BEL
H	6B	h	ØB	BS	4B	H	ØB	BS
I	69	i	Ø9	HT	49	I	Ø9	HT
J	6A	j	ØA	LF	4A	J	ØA	LF

KEY	NORMAL	CHAR	CONTROL	CHAR	SHIFT	CHAR	BOTH	CHAR
K	6B	k	ØB	VT	4B	K	ØB	VT
L	6C	l	ØC	FF	4C	L	ØC	FF
M	6D	m	ØD	CR	4D	M	ØD	CR
N	6E	n	ØE	SD	4E	N	ØE	SQ
Q	6F	o	ØF	SI	4F	Q	ØF	SI
P	7Ø	p	1Ø	DLE	5Ø	P	1Ø	DLE
Q	71	q	11	DC1	51	Q	11	DC1
R	72	r	12	DC2	52	R	12	DC2
S	73	s	13	DC3	53	S	13	DC3
T	74	t	14	DC4	54	T	14	DC4
U	75	u	15	NAK	55	U	15	NAK
V	76	v	16	SYN	56	V	16	SYN
W	77	w	17	ETB	57	W	17	ETB
X	7B	x	1B	CAN	5B	X	1B	CAN
Y	79	y	19	EM	59	Y	19	EM
Z	7A	z	1A	SUB	5A	Z	1A	SUB
Ø	3Ø	Ø	3Ø	Ø	3Ø	Ø	3Ø	Ø
1	31	1	31	1	31	1	31	1
2	32	2	32	2	32	2	32	2
3	33	3	33	3	33	3	33	3
4	34	4	34	4	34	4	34	4
5	35	5	35	5	35	5	35	5
6	36	6	36	6	36	6	36	6
7	37	7	37	7	37	7	37	7
B	3B	B	3B	B	3B	B	3B	B
9	39	9	39	9	39	9	39	9
.	2E	.	2E	.	2E	.	2E	.
+	2B	+	2B	+	2B	+	2B	+
–	2D	–	2D	–	2D	–	2D	–
*	2A	*	2A	*	2A	*	2A	*
/	2F	/	2F	/	2F	/	2F	/
PAUSE	13	DC3	13	DC3	13	DC3	DC13	DC3
BREAK	Ø3	ETX	Ø3	ETX	Ø3	ETX	Ø3	ETX
ENTER	ØD	CR	ØD	CR	ØD	CR	ØD	CR
F1	ØØ	NUL	ØØ	NUL	ØØ	NUL	ØØ	NUL
F2	Ø1	SQH	Ø1	SQH	Ø1	SQH	Ø1	SQH
F3	Ø2	STX	Ø2	STX	Ø2	STX	Ø2	STX
F4	Ø3	ETX	Ø3	ETX	Ø3	ETX	Ø3	ETX
F5	Ø4	EQT	Ø4	EQT	Ø4	EQT	Ø4	EQT
F6	Ø5	ENQ	Ø5	ENQ	Ø5	ENQ	Ø5	ENQ
F7	Ø6	ACK	Ø6	ACK	Ø6	ACK	Ø6	ACK
FB	Ø7	BEL	Ø7	BEL	Ø7	BEL	Ø7	BEL
F9	ØC	FF	ØC	FF	ØC	FF	ØC	FF
F1Ø	1B	CAN	1B	CAN	1B	CAN	1B	CAN

APPENDIX F

DISPLAY CHARACTERS

F. DISPLAY CHARACTER

	INVERSE				FLASHING				NORMAL							
	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
+$0	@	P		0	@	P		0	@	P		0	@	P	`	p
+$1	A	O	!	1	A	O	!	1	A	O	!	1	A	O	a	q
+$2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	b	r
+$3	C	5	#	3	C	S	#	3	C	5	#	3	C	5	c	s
+$4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	d	t
+$5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	e	u
+$6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	f	v
+$7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	g	w
+$8	H	X	(	8	H	X	(	8	H	X	(	B	H	X	h	x
+$9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	i	y
+$A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	j	z
+$B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	k	{
+$C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	l	¦
+$D	M	]	–	=	M	]	–	=	M	]	–	=	M	]	m	}
+$E	N	^	.	>	N	^	.	>	N	^	.	>	N	^	n	~
+$F	O	–	/	?	O	–	/	?	O	–	/	?	O	–	o	▩

Table F-1 Primary display character set

	INVERSE				MOUSE		INVERSE		NORMAL							
	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
+$0	@	P		0	▲	↖	`	p	@	P		0	@	P	`	p
+$1	A	O	!	1	△	↗	a	q	A	O	!	1	A	O	a	q
+$2	B	R	"	2	▲	↘	b	r	B	R	"	2	B	R	b	r
+$3	C	5	#	3	⧖	↙	c	s	C	5	#	3	C	5	c	s
+$4	D	T	$	4	✓	⌐	d	t	D	T	$	4	D	T	d	t
+$5	E	U	%	5	◥	→	e	u	E	U	%	5	E	U	e	u
+$6	F	V	&	6	↑	▩	f	v	F	V	&	6	F	V	f	v
+$7	G	W	'	7	↑	▩	g	w	G	W	'	7	G	W	g	w
+$8	H	X	(	B	↑	⌐	h	x	H	X	(	8	H	X	h	x
+$9	I	Y	)	9	↓	⌐	i	y	I	Y	)	9	I	Y	i	y
+$A	J	Z	*	:	↑	◆	j	z	J	Z	*	:	J	Z	j	z
+$B	K	[	+	;	↑	▪	k	{	K	[	+	;	K	[	k	{
+$C	L	\	,	<	↑	▪	l	▌	L	\	,	<	L	\	l	▌
+$D	M	]	–	=	◥	▪	m	}	M	]	–	=	M	]	m	}
+$E	N	^	.	>	◥	▪	n	~	N	^	.	>	N	^	n	~
+$F	O	–	/	?	◥	▪	o	▩	O	–	/	?	O	–	o	▩

Table F-2 Alternate display character set

APPENDIX G

ASCII CHARACTER CODES

G. ASCII CHARACTER CODES

The following table lists all the ASCII codes and their associated characters. These characters can be displayed using PRINT CHR$(n), where n is the ASCII code. ASCII codes 0 to 31 are control characters (usually used for control functions or communications). They are all non-printing characters.

ASCII			ASCII		
DECIMAL	HEX	CHARACTER	DECIMAL	HEX	CHARACTER
000	00	NUL	027	1B	ESC
001	01	SOH	028	1C	FS
002	02	STX	029	1D	GS
003	03	ETX	030	1E	RS
004	04	EOT	031	1F	US
005	05	ENQ	032	20	(space)
006	06	ACK	033	21	!
007	07	BEL	034	22	"
008	08	BS	035	23	#
009	09	HT	036	24	$
010	0A	LF	037	25	%
011	0B	VT	038	26	&
012	0C	FF	039	27	'
013	0D	CR	040	28	(
014	0E	SO	041	29	)
015	0F	SI	042	2A	*
016	10	DLE	043	2B	+
017	11	DC1	044	2C	,
018	12	DC2	045	2D	—
019	13	DC3	046	2E	-
020	14	DC4	047	2F	/
021	15	NAK	048	30	0
022	16	SYN	049	31	1
023	17	ETB	050	32	2
024	18	CAN	051	33	3
025	19	EM	052	34	4
026	1A	SUB	053	35	5

DECIMAL	HEX	CHARACTER	DECIMAL	HEX	CHARACTER
054	36	6	091	5B	[
055	37	7	092	5C	\
056	38	8	093	5D	]
057	39	9	094	5E	^
058	3A	:	095	5F	—
059	3B	;	096	60	'
060	3C	<	097	61	a
061	3D	=	098	62	b
062	3E	>	099	63	c
063	3F	?	100	64	d
064	40	@	101	65	e
065	41	A	102	66	f
066	42	B	103	67	g
067	43	C	104	68	h
068	44	D	105	69	i
069	45	E	106	6A	j
070	46	F	107	6B	k
071	47	G	108	6C	l
072	48	H	109	6D	m
073	49	I	110	6E	n
074	4A	J	111	6F	o
075	4B	K	112	70	p
076	4C	L	113	71	q
077	4D	M	114	72	r
078	4E	N	115	73	s
079	4F	O	116	74	t
080	50	P	117	75	u
081	51	Q	118	76	v
082	52	R	119	77	w
083	53	S	120	78	x
084	54	T	121	79	y
085	55	U	122	7A	z
086	56	V	123	7B	{
087	57	W	124	7C	|
088	58	X	125	7D	}
089	59	Y	126	7E	-
090	60	Z	127	7F	DEL

APPENDIX H

MATHEMATICAL FUNCTIONS

H. MATHEMATICAL FUNCTIONS

Functions that are not intrinsic to Personal Computer BASIC may be calculated as follows.

Function	Equivalent
Secant	SEC(x)=1/COS(x)
Cosecant	CSC(x)=1/SIN(x)
Cotangent	COT(x)=1/TAN(x)
Inverse sine	ARCSIN(x)=ATN(x/SQR(1-x*x))
Inverse cosine	ARCCOS(x)=1.570796-ATN (x/SQR(1-x*x))
Inverse secant	ARCSEC(x)=ATN(1/AQR (x*x-1))+(x<0)*3.141593
Inverse cosecant	ARCCSC(x)=ATN((1/SQR (x*x-1))+(x<0)*3.141593
Inverse cotangent	ACCOT(x)=1.57096-ATN(x)
Hyperbolic sine	SINH(x)=(EXP(x)-EXP-x))/2
Hyperbolic cosine	COSH(x)=(EXP(x)+ EXP(-x))/2
Hyperbolic tangent	TANH(x)=EXP(x)-EXP (-x))/(EXP(x)+EXP(-x))
Hyperbolic secant	SECH(x)=2/(EXP(x)+EXP(-x))
Hyperbolic cosecant	CSCH(x)=2/(EXP(x)-EXP(-x))
Hyperbolic cotangent	COTH(x)=(EXP(-x))/(EXP (x)-EXP(-x))

Function	Equivalent
Inverse hyperbolic sine	ARCSINH(x)=LOG (x+SQR(x*x+1))
Inverse hyperbolic cosine	ARCCOSH(x)=LOG (x+SQR(x*x+-1))
Inverse hyperbolic tangent	ARCTANH(x)=LOG ((1+x)/(1-x))/2
Inverse hyperbolic secant	ARCSECH(x)=LOG ((1+SQR(1-x*x))/x)
Inverse hyperbolic cosecant	ARCCSCH(x)=LOG ((1+SGN(x)*SGN(x)*SQR (1+x*x))/x)
Inverse hyperbolic cotangent	ARCCOTH(x)=LOG ((x+1)/(x-1))/2

If you use these functions, a good way to code them would be using the DEF FN statement. for example, instead of typing the formula for inverse hyperbolic sine each time you need it, you could use a program line.

] 10 DEF FN INSIH(x) = LOG (x * x + 1)

, then refer to it as

] 20 Z = FN INSIH(x)

APPENDIX I

SUMMARY OF BASIC COMMANDS

I. SUMMARY OF BASIC COMMANDS

COMMAND	DESCRIPTION
ABS	To give the absolute value of a numeric expression.
AMPERSAND	To jump into a machine language command starting at hex location $3F5.
ASC	To return the ASCII code of the first character of the specified string.
ATN	To calculate the arctangent of a value.
CALL	to use an assembly language subroutine.
CHR$	Converts an ASCII code to its equivalent character.
CLEAR	To clear all variables, arrays and strings.
COLOR	To set the color of subsequently plotted low resolution graphics.
CONT	To start a program running again after it has been halted.
COS	To calculate the cosine of an angle.
DATA	To store constant numbers and string values in your program so they can be used in conjunction with the READ statement.

DEF FN	Allows you to define and name a function.
DEL	Removes program lines.
DIM	This gives the values for the subscripts of arrays, and allocates enough storage to accomodate them.
DRAW	To draw pre-defined geometric shapes.
END	Finishes program execution and returns you to command level.
EXP	To calculate the value of "e" - the base of natural logarithms - raised to a specified power.
FLASH	To cause all computer messages to alternate between character and background colour.
FOR...NEXT	Loops around a group of instructions a specified number of times.
FRE	Reports on the number of bytes in memory that are not being used by BASIC.
GET	Reads a character from the keyboard without echoing it on the screen. No carriage return is necessary.
GOSUB...RETURN	To direct the program flow into, and return from, a subroutine.

GOTO	To direct the program flow to another part of a BASIC program.
GR HGR HGR2	To set up the different graphics modes.
HCOLOR	To set the color of subsequently plotted high resolution graphics.
HIMEM:	To set the highest memory location available to a BASIC program.
HLIN	To draw a horizontal line in low resolution graphics.
HOME	To clear screen and position the cursor to the upper left corner of the display screen.
HPLOT	To draw either lines or dots in high resolution graphics.
HTAB	To move the cursor a given number of places to the right of the left margin.
IF...GOTO and	To direct program flow depending on the result of an evaluation.
IN#	To accept input from selected input device.
INPUT	Allows you to enter values from the keyboard while a program is executing.
INT	To round a fractional number down to a whole number.
INVERSE	To reverse the character and background color of the video display.

LEFT$	Returns a specified number of characters from the left-hand-side of a character string.
LEN	To return the number of characters in a string.
LET	To assign a value to a variable.
LIST	To display on the screen the BASIC program that is currently in memory.
LOG	To calculate the natural logarithm of a specified value.
LOMEM:	To set the lowest memory location available to a BASIC program.
MID$	To return a specified number of characters from within a given string.
NEW	Clears the current program from memory and clears all variables associated with it.
NORMAL	To return the video display from either inverse or flashing modes to the default mode.
NOTRACE	To stop program statement numbers from being displayed as a program is executed.
ON...GOSUB	To direct the program flow depending on the value of an expression.
ONERR GOTO	To avoid halting the program when an error is encountered.

PDL	To return the current value of the game adapter.
PEEK	To read the byte at a specified memory location.
PLOT	To draw dots in low resolution graphics.
POKE	To write a byte of data into a specified memory location.
POP	To change the action of a RETURN from a subroutine.
POS	To return the current horizontal cursor position.
PR#	To switch the output to the selected device.
PRINT	To display characters on the display screen.
READ	To read values form a DATA statement and to allocate them to variables.
REM	To let you REMind yourself by REMarks of what your program is doing.
RESTORE	To use DATA values again after they have been READ.
RESUME	To restart a program that has been halted due to an error.
RETURN	To return execution to the line immediately following the most recent GOSUB statement.

RIGHT$	To return a specified number of characters from a string proceeding from the right.
RND	To return a random number between 0 and 1.
ROT	To specify the angle by which a shape is rotated when drawn on the screen, used in conjunction with DRAW or XDRAW.
RUN	To start a program execution.
SCALE	To increase or decrease the size of shapes created by DRAW or XDRAW.
SCRN	To give the color code of a point in low resolution graphics.
SGN	To return the sign of a number.
SIN	To calculate the sine of a specified angle.
SPC	To separate two printed items by a specified number of spaces.
SPEED	To specify the rate at which characters are to be sent to an output device.
SQR	To calculate the square root of a specified value.
STOP	To halt a program execution and return to command level.
STR$	To return a string representation of a numeric value.

TAB	To move the cursor a specified number of places to the right of the left margin.
TEXT	To set the display to full-screen text mode.
TRACE	To display line numbers of a program as it is being executed.
USR	This command specifies a parameter of an assembly language subroutine.
VAL	To return the numerical value of a specified string.
VLIN	To draw a vertical line in low resolution graphics.
VTAB	To move the cursor a given number of lines down the display screen.
WAIT	To suspend a program's execution while monitoring the status of an input port.
XDRAW	To draw or erase a defined shape.

APPENDIX J

LIST OF RESERVED WORDS IN BASIC

J. LIST OF RESERVED WORDS IN BASIC

LIST OF RESERVED WORDS IN BASIC

ABS	GR	NOTRACE	SPC (
AND	HCOLOR =	ON	SPEED =
ASC	HGR	ONERR	SQR
AT	HGR2	OR	STEP
ATN	HIMEM:	PDL	STOP
CALL	HLIN	PEEK	STR$
CHR$	HOME	PLOT	TAB (
CLEAR	HPLOT	POKE	TAN
COLOR =	HTAB	POP	TEXT
CONT	IF	POS	THEN
COS	IN#	PRINT	TO
DATA	INPUT	PR#	TRACE
DEF	INT	READ	USR
DEL	INVERSE	REM	VAL
DIM	LEFT$	RESTORE	VLIN
DRAW	LEN	RESUME	VTAB
END	LET	RETURN	WAIT
EXP	LIST	RIGHT$	XDRAW
FLASH	LOG	RND	&
FN	LOMEM:	ROT =	+
FOR	MID$	RUN	-
FRE	NEW	SCALE =	*
GET	NEXT	SCRN (	/
GOSUB	NORMAL	SGN	>
GOTO	NOT	SIN	<
			=
			^

IBM-International Business Machines Corp.
Apple-Apple Computer Inc..
Microsoft-Microsoft Corp..

 91-2104-10-00